Fachhochschule Köln
University of Applied Sciences Cologne
Abteilung Gummersbach
Fakultät für Informatik und Ingenieurwissenschaften

Diplomarbeit

Zur Erlangung
des Diplomgrades
Diplom-Informatiker (FH)
in der Fachrichtung Wirtschaftsinformatik

Thema

- Entwicklung einer Web-Applikation zur Archivierung und Auswertung statistischer Daten eines Automatisierungssystems unter Verwendung des Frameworks Apache Cocoon -

Erstprüferin:	Prof. Dr. Heide Faeskorn-Woyke
Zweitprüfer:	Prof. Dr. Holger Günther
vorgelegt	April 2004
von	Uwe Stelzer Bruchhausener Str. 8 51702 Bergneustadt
Tel.- Nr:	02261 479760
Email:	UweStelzer@web.de
Matr.-Nr.:	11025775

Autor: Uwe Stelzer Matr.-Nr. 11025775

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

API	Application Programming Interface
ARP	Address Resolution Protocol
AWT	Abstract Window Toolkit
CASE	Computer Aided Software Engineering
CGI	Common Gateway Interface
CHAP	Challenge Handshake Authentication Protocol
CSV	Comma Separated Values
CVS	Concurrent Versions System
DBMS	Data Base Management System
DHCP	Dynamic Host Configuration Protocol
DOM	Document Object Model
ERD	Entity Relationship Diagram
FDDI	Fiber Distributed Data Interface
FOP	Formatting Objects Processor
FTP	File Transfer Protocol
GNU	GNU's not Unix
HMI	Human Machine Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IDE	Integrated Development Environment
ISDN	Integrated Services Digital Network
JDBC	Java Database Connectivity
JDK	Java Development Kit
JPG, JPEG	Joint Photographic Experts Group
JRE	Java Runtime Environment
Modem	Modulator/Demodulator
NetBIOS	Network Basic Input/Output System
ODBC	Open Database Connectivity
PAP	Password Authentication Protocol

PDF	Portable Document Format
PNG	Portable Network Graphics
PPP	Point to Point Protocol
pppd	Point to Point Protocol Daemon
RARP	Reverse Address Resolution Protocol
RAS	Remote Access Service
RPM	Red Hat Package Manager
SAX	Simple API for XML
SDK	Software Development Kit
SMB	Server Message Block
SMTP	Simple Mail Transport Protocol
SoC	Separation of Concerns
SQL	Structured Query Language
SVG	Scalable Vector Graphics
TCP/IP	Transmission Control Protocol / Internet Protocol
UDP	User Datagram Protocol
UML	Unified Modelling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W-LAN	Wireless Local Area Network
WinCC	Windows Control Center
XML	eXtensible Markup Language
XSL	eXtensible Stylesheet Language
XSLT	eXtensible Stylesheet Language Transformations
XSP	eXtensible Server Pages

1 Einleitung

In diesem Kapitel wird zur Einführung die Firma SMS Eumuco und ihr Tätigkeitsfeld vorgestellt. Dann wird die Thematik, in deren Kontext die Aufgabenstellung der Diplomarbeit eingebettet ist, beschrieben. Daraufhin folgt eine Beschreibung der Zielsetzung der Diplomarbeit.

1.1 Unternehmensbeschreibung

Die SMS Eumuco GmbH ist ein Maschinenbauunternehmen, das in der Branche der Press- und Schmiedetechnik tätig ist. Sie ist ein Tochterunternehmen der SMS- Gruppe im Unternehmensbereich Schmiedetechnik. Die SMS-Gruppe gliedert sich in die Unternehmensbereiche Hütten- und Walzwerkstechnik, Rohr-, Profil- und Schmiedetechnik sowie Kunststofftechnik.

Die SMS Aktiengesellschaft ist die geschäftsführende Holding und verantwortlich für die Führung und Kontrolle der Unternehmensgruppe.

Die Wurzeln der SMS Eumuco GmbH reichen bis in die Anfänge des Maschinenbaus zurück. Sie ist hervorgegangen aus den Unternehmen Schloemann, Eumuco, Hasenclever, Wagner Dortmund und Banning.

Das Leistungsspektrum gliedert sich in vier Produktbereiche:

1) Produktbereich Schloemann Strangpressen

Der Unternehmensbereich Strangpressen fertigt Strangpressanlagen für die Leichtmetallverarbeitung. Das Programm umfasst alles von den Strangpressen bis zu automatischen Auslauf- und Handling-Einrichtungen, Ofenausrichtungen, Oberflächenbeschichtungsanlagen oder Produktionssteuerungs- und Überwachungssystemen.

Strangpressprofile werden zum Beispiel in Gebäudefassaden oder im modernen Fahrzeugbau verwendet.

2) Produktbereich Wagner Banning Ringwalzen

Dieser Unternehmensbereich konstruiert und baut Ringwalzanlagen zur Fertigung von ring- und räderförmigen Produkten. Das Programm umfasst Ringwalzmaschinen und -anlagen für die Herstellung ringförmiger Werkstücke von 100 bis 8.000 mm Durchmesser und Größe bei Ringhöhen bis zu 1.800 mm und höher.

Die Anwendungsbereiche für diese nahtlos gewalzten Ringe sind zum Beispiel Automobilkomponenten, Eisenbahnkomponenten und Transportsysteme.

3) Produktbereich Blankstahlanlagen

Dieser Unternehmensbereich stellt Anlagentechnik zur Kaltveredelung von Stabstahl her. Zur Kaltveredelung zählen Arbeitsgänge wie Schälen, Richten, Polieren, Fasen, Schleifen und mehr.

Besonders gefordert ist Blankstahl im Automobilbau, zum Beispiel für Schrauben, Kolbenstangen, Zündkerzen, Ventile, Stoßdämpferstangen, Walzkörper, Schlossteile und Stabilisatoren.

4) Produktbereich Eumuco Hasenclever Gesenkschmieden

Dieser Produktbereich umfasst Gesenkschmieden und komplette Gesenkschmiedeanlagen für die unterschiedlichsten Werkstoffe. Gesenkschmieden ist ein Umformen in zwei gegeneinander bewegten Werkzeughälften – den Gesenken. Der Gesenkhohlraum entspricht der Form des herzustellenden Gesenkschmiedeteils.

Die so gefertigten Schmiedeteile werden überall dort eingesetzt, wo es um Sicherheit, Zuverlässigkeit und Lebensdauer geht: in der Automobilindustrie, im Nutzfahrzeugbau, Flugzeugbau, Turbinenbau, Eisenbahnwesen, Bergbau und in der Landwirtschaft.

Der Umsatz der SMS Eumuco GmbH lag 2001 (2002) bei 139 (150) Millionen €. In diesem Jahr waren im Unternehmensbereich durchschnittlich 538 (570) Mitarbeiter beschäftigt, davon ca. 110 (130) in den Montage- und Fertigungsbereichen. [SMS02], [SMS03]

1.2 Beschreibung der Thematik

Moderne Schmiedeanlagen sind heutzutage zwecks wirtschaftlicherer Produktion kom-

plett automatisiert. Zu ihrer Steuerung werden Automatisierungssysteme eingesetzt. Zu diesen Systemen gehören verschiedene Komponenten wie Tasten- oder Zeilenpanels und Steuertafeln, die über ein Bussystem miteinander verbunden sind. Diese Systeme ermöglichen die genaue Auswertung des Anlagenzustandes, eine hohe Verfügbarkeit durch den Einsatz von Diagnoseprogrammen und eine umfassende Prozesssteuerung und -kontrolle.

Die Anlagen der Firma SMS Eumuco werden von dem Automatisierungssystem SIMATIC der Firma Siemens gesteuert und überwacht. Zu diesem Automatisierungssystem gehört das HMI-System (Human Machine Interface) WinCC. WinCC ist auf einem Industrie-PC mit dem Betriebssystem Windows 2000 vor Ort installiert. Es ist über den Prozessbus PROFIBUS mit den Steuergeräten des Automatisierungssystems verbunden. WinCC ermöglicht die Visualisierung und Bedienung der Maschinenprozesse sowie die Alarmierung bei Fehlerzuständen. Mit WinCC können grafische Bedienoberflächen zur Prozessvisualisierung und -steuerung erstellt werden (Projektierung). Es ermöglicht die Archivierung der Meldungen und Prozesswerte für den späteren Zugriff [Sie01]. Abbildung 1.1 zeigt die schematische Übersicht einer Schmiedeanlage in WinCC während der Laufzeit des Visualisierungssystems. Gezeigt wird die logische Anordnung der Maschinen von der Vorerwärmung links oben bis zur eigentlichen Presse rechts unten.

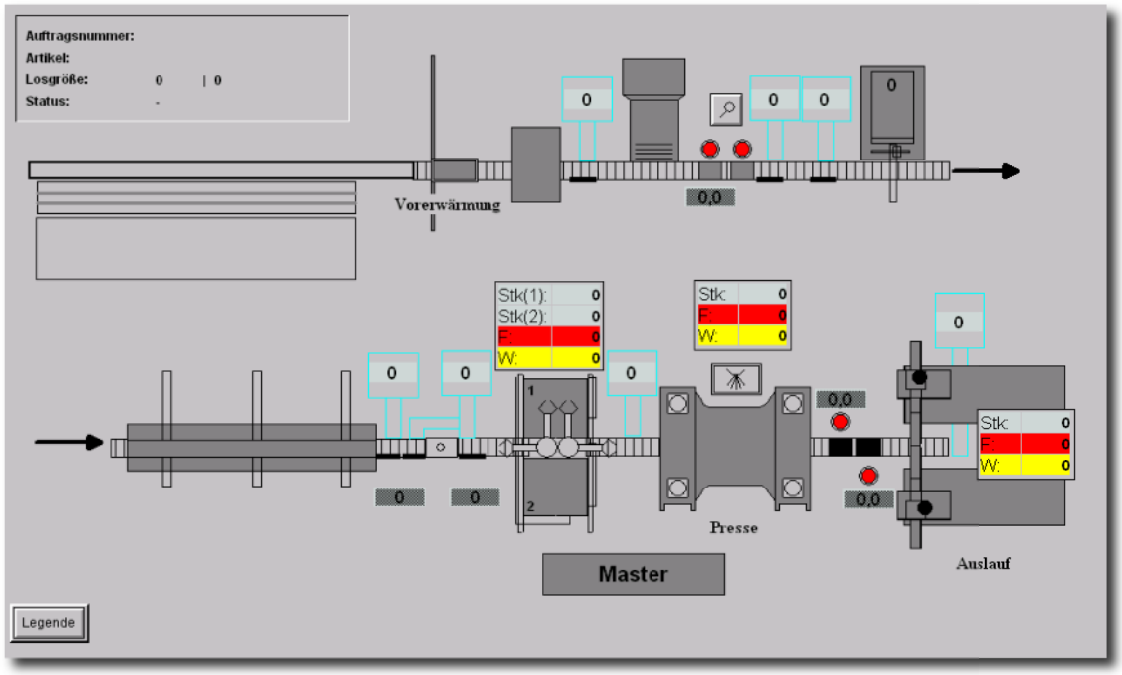

Abbildung 1.1: Anlagenübersicht eines Visualisierungsprojektes mit WinCC

1.3 Zielsetzung

Das Produkt soll es den Ingenieuren der Konstruktionsabteilung von SMS Eumuco ermöglichen, Messwerte, Produktions- und Betriebsdaten von den bei den Kunden installierten Gesenkschmiedeanlagen lückenlos zu erfassen, und regelmäßig zu archivieren und auszuwerten. Dabei soll eine Kopie der auf dem HMI-System des Kunden gespeicherten Konfigurations- und Betriebsdaten in einer relationalen Datenbank abgelegt werden, die zu diesem Zweck auf einem Rechner im Hausnetz von SMS Eumuco angelegt wird. Die Datenbank soll dabei ein Backupsystem für die Betriebsdaten darstellen, das auch im Störungsfall oder bei Ausfall des HMI-Systems der Firma Eumuco einen Zugriff auf diese Daten ermöglicht. Als Datenbankmanagementsystem ist Oracle 9i für Linux vorgesehen.

Da die Betriebsdaten laufend ergänzt werden, ist eine laufende Aktualisierung und Übertragung dieser Daten erforderlich.

Das Produkt soll eine Funktionalität bieten, die bisher durch eine behelfsmäßige, manuelle Lösung bei einer Schmiedeanlage realisiert wird:

Für die langfristige Archivierung und Auswertung wird von Eumuco-Mitarbeitern eine Verbindung zum WinCC-PC mit der Remote-Control-Software pcAnywhere aufgebaut. Zu diesem Zwecke wird eine Modemverbindung genutzt, da der PC weit entfernt bei der Schmiedeanlage des Kunden steht. Die gewünschten Daten werden mithilfe eines ANSI-C-Skripts aus dem WinCC-Modul „Global Script“ als flache Textdateien im CSV-Format exportiert. Dieser Exportvorgang wird einmal täglich durch einen Trigger ausgelöst. Die Textdateien werden dann mit der Dateiübertragungsfunktion von pcAnywhere von Hand kopiert. Für die Übertragung werden alle Dateien automatisch von pcAnywhere komprimiert. Nach der Übertragung werden sie mit einem in Visual Basic geschriebenen Programm mit Komponenten der Reporting-Software „Crystal Reports“ ausgewertet.

1.3.1 Datenübertragung

Die Daten der Anlage sollen also von dem WinCC-System vor Ort zu einem Datenbankserver im Hause Eumuco übertragen werden. Die Übertragung soll automatisiert erfolgen. Eine Verbindungsmöglichkeit zu dem WinCC-PC der Anlage besteht nur über Modem. Das präferierte Übertragungsmedium sind die Textdateien, die durch die Exportskripts von WinCC ja bereits erstellt werden.

Deshalb soll eine Anwendung erstellt werden, die eine Verbindung zum WinCC-PC aufbaut, über ein geeignetes Dateiübertragungsprotokoll die Textdateien überträgt und diese in eine SQL-Datenbank importiert. Das Ausführen dieser Anwendung soll automatisiert einmal täglich erfolgen – zu einem Zeitpunkt, nachdem die tagesaktuellen Textdateien erstellt worden sind.

1.3.2 Auswertung

Die Auswertung der gewonnenen Daten sollte folgende Kriterien erfüllen:

- Es soll eine visuelle Darstellung in Form von Datenbank-Reports mit Tabellen und Diagrammen möglich sein.
- Die Reports sollen in einer druckbaren Form erzeugbar sein.
- Produktionsdaten sollen tabellarisch dargestellt werden.
- Betriebsdaten sollen als Protokolle dargestellt werden.

Zusammenfassend gesagt ist die Zielsetzung die Konzeption eines angepassten Reporting-Systems auf Basis einer leistungsfähigen Datenbank und mit einem automatisierten Datenimportmechanismus.

2 Grundlagen

In diesem Kapitel werden die Softwarekonzepte beschrieben, die für das Grundverständnis der Arbeitsweise der während der Diplomarbeit entwickelten Anwendung notwendig sind. Hierzu wird eine Definition der Begriffe Web-Applikationen und Frameworks gegeben.

2.1 Web-Applikationen

Man bezeichnet im Allgemeinen viele sehr unterschiedliche Computerprogramme als Web-Applikationen, die im Wesentlichen alle eine Beziehung zum Internet gemeinsam haben. Web-Applikationen kommunizieren über das Internet, werden über das Internet verbreitet oder sind über mehrere Rechner im Netzwerk verteilt.

Die Art von Web-Applikation, die für dieses Projekt erstellt werden soll, entspricht einer klassischen Erscheinungsform: der 3-Tier Web-Applikation. Eine 3-Tier[1] Anwendung besteht aus drei diskreten Stufen:

- **First Tier**
  Diese Stufe leistet die Darstellung einer Benutzeroberfläche und wird in der Regel durch einen Webbrowser dargestellt.
- **Second (Middle) Tier**
  Diese Stufe enthält die Anwendungslogik zur Erstellung von dynamischen Inhalten. Sie wird meistens durch CGI-Skripts oder Servlets realisiert.
- **Third Tier**
  Diese Stufe dient der persistenten Speicherung von Daten, auf die die zweite Stufe zugreift oder die vom Benutzer generiert werden. Oft wird diese Stufe durch eine Datenbank realisiert.

Das Zusammenspiel der drei Stufen erfolgt folgendermaßen: Der Webbrowser sendet eine Anfrage an die Middle Tier, die sie beantwortet indem sie eine Abfrage an die Datenbank erzeugt, eine Benutzeroberfläche generiert und diese an den Webbrowser als Antwort zurückschickt.

1 englisch „Rang, Stufe“

Eine besondere Eigenschaft dieser Architektur ist, dass sie verteilt sein kann - das heißt, jede der drei Stufen kann auf einem eigenen Rechner installiert sein. Die Kommunikation zwischen den Stufen erfolgt dann über das zugrunde liegende Netzwerk. Dies kann ein firmeninternes Netzwerk oder auch das Internet sein.

2.1.1 Vorteile

Ein wichtiger Vorteil von Web-Applikationen gegenüber anderen Applikationen ist die Verfügbarkeit. Durch die Ubiquität von Webbrowsern muss eine Web-Applikation nicht installiert werden sondern kann an jedem netzwerkfähigen Arbeitsplatz ausgeführt werden. Auch sind sie aufgrund dieses Umstandes betriebssystemunabhängig, zumindest dann, wenn man sich bei der Entwicklung nicht auf einen Browser spezialisiert.

Durch die Trennung von Bedienoberfläche und „Engine“ sind eine zentrale Konfiguration und schnelle Updates möglich, was den Arbeitsaufwand insbesondere bei vielen Clients verringert.

Web-Applikationen sind durch ihre Client-Server-Architektur in der Regel mehrbenutzerfähig, das heißt, dass mehrere Benutzer gleichzeitig mit derselben Anwendung und derselben Datenbasis arbeiten können.

2.1.2 Nachteile

Die Oberfläche eines Webbrowsers ist vor allem durch die geringe Anzahl an Bedienelementen die vom HTML-Standard vorgegeben sind funktional stark eingeschränkt gegenüber nativen grafischen Oberflächen.

Bei der professionellen Entwicklung von Web-Applikationen müssen verschiedene Entwicklergruppen (Programmierer, Webdesigner, Datenbankspezialisten, Administratoren) koordiniert werden.

Durch die räumliche Trennung der Stufen müssen viele Daten über Netzwerkverbindungen transportiert werden. Bei langsamen Verbindungen besonders über das Internet kann dies die Arbeitsgeschwindigkeit negativ beeinflussen.

2.2 Frameworks

Der sehr allgemeine Begriff Framework[2] lässt schon vermuten, dass damit keine konkrete Art von Applikationen beschrieben werden soll. Es handelt sich also nicht einfach um Emailprogramme, Datenbankanwendungen, Javaprogramme, Officepakete[3] oder ähnliche Kategorien von Applikationen.

Vielmehr handelt es sich bei einem Framework um eine Sammlung von Softwarebibliotheken, die die Anwendungsarchitektur einer Applikation festlegen und zum Teil bereits implementieren. Sie stellen eine generische Lösung für häufig wiederkehrende Aufgabenstellungen bereit. In der objektorientierten Programmierung stellen Frameworks abstrakte Klassen bereit, von denen dann die konkreten Klassen für die jeweilige Anwendungsschicht abgeleitet werden [Oes01].

Frameworks bieten eine gute Möglichkeit unnötigen Programmieraufwand zu vermeiden, der durch das häufige Neuprogrammieren von sich ähnelndem Programmcode entstehen kann.

2.2.1 Softwarebibliotheken

Das nicht nur in der Informatik bekannte Problem der „Neuerfindung des Rades“ wird bereits seit den Urzeiten der Computerprogrammierung durch Bildung von Softwarebibliotheken (auch Programmbibliotheken) zu vermeiden versucht. Bibliotheken bieten Algorithmen für zusammengehörende Aufgaben. Sie können als Quelltext oder als statisch beziehungsweise dynamisch gebundene Programmbibliothek vorliegen.

Statische Programmbibliotheken werden nach dem Kompilieren in einem zusätzlichen Schritt, dem so genannten Linken mit einem ausführbaren Programm verbunden.

Dynamische Programmbibliotheken werden erst bei Bedarf in den Arbeitsspeicher geladen und dann mit einem ausführbaren Programm verbunden. Dadurch braucht eine Programmbibliothek, die von mehreren Programmen genutzt wird, auch nur einmal im Speicher gehalten werden.

2 englisch „Rahmenwerk“
3 Obwohl kurioserweise gerade unter der generischen Adresse http://www.framework.com/ ein Officepaket unter eben diesem Namen angeboten wird.

In jedem Fall nutzt das ausführbare Programm Methoden, Daten oder Objekte der Bibliothek. Diese müssen sehr allgemein gültig und mit besonderem Augenmerk auf Wiederverwendbarkeit programmiert worden sein.

Der Programmierer einer Anwendung, die eine Bibliothek benutzt, muss keinerlei Kenntnisse darüber besitzen, wie die Algorithmen der Bibliothek funktionieren. Er benötigt lediglich das Wissen um die Schnittstellen, über die er auf die Funktionalität zugreift – das können zum Beispiel Namen und Datentypen von Objekten, Methoden und Parametern sein. In der objektorientierten Programmierung nennt man das Verstecken der Implementierungsdetails hinter Schnittstellen „Kapselung“.

2.2.2 Architekturen

Frameworks erweitern das Konzept der Softwarebibliotheken: Sie bieten nicht nur Programmcode für einzelne Aufgaben, sondern sie geben einer Anwendung eine feste Anwendungsarchitektur vor. Ein Framework beschreibt also die Architektur eines bestimmten Typs von Anwendungen.

Dabei implementiert es bereits solche Funktionen, die für alle Anwendungen dieses Typs gleich sind, also immer wieder benötigt werden. Dies sind die Kernkomponenten eines Frameworks. Je weiter ein Framework entwickelt ist, desto weniger Programmieraufwand ist für die Erstellung der darauf basierenden Anwendung von Nöten. Im Extremfall muss ein Framework sozusagen nur noch „konfiguriert“ werden, um eine einsatzbereite Anwendung zu realisieren.

Der Programmierer einer Anwendung, die auf ein Framework aufsetzt, muss zusätzlich zu den Schnittstellen die Architektur des Frameworks erlernen, um damit eine Anwendung erstellen zu können. In objektorientierten Frameworks wird eine solche Architektur zum Beispiel durch zusammenhängende Basisklassen vorgegeben, von denen eigene Klassen abgeleitet (vererbt) werden müssen. Diese entsprechen dadurch automatisch einem Schema, das die Architektur festlegt.

2.2.3 Ein Beispiel mit Schnittstellen und Vererbung

Ein zusätzliches Konzept zur Modellierung von Architekturen bietet die Programmier-

sprache Java mit den so genannten Interfaces (Schnittstellen). Eine Javaschnittstelle ist eine abstrakte Klasse, die keine Implementierungen, sondern nur Namen und Signaturen der enthaltenen Methoden vorgibt – im Prinzip also die Architektur einer einzelnen Klasse [Ull03]. Eine Klasse, die die Schnittstelle verwenden möchte, muss dies durch das Schlüsselwort `implements` in ihrer Deklaration anzeigen und dann die Methoden der Schnittstellen implementieren.

Schnittstellen stellen somit im Prinzip einen alternativen Mechanismus für den Spezialfall der Vererbung von *abstrakten* Basisklassen dar.

Ein Beispiel für Basisklassen, realisiert mit Vererbung von abstrakten Klassen und Schnittstellen in Java zeigt das UML-Diagramm in Abbildung 2.1. Das Beispiel enthält ein kleines Szenario, das eine Vererbungsbeziehung zwischen Kuh und Säugetier mittels Schnittstellenimplementation (gestrichelter Pfeil, <<realize>>) darstellt und eine andere Vererbungsbeziehung zwischen Gras und Nahrung mittels Vererbung einer abstrakten Basisklasse.

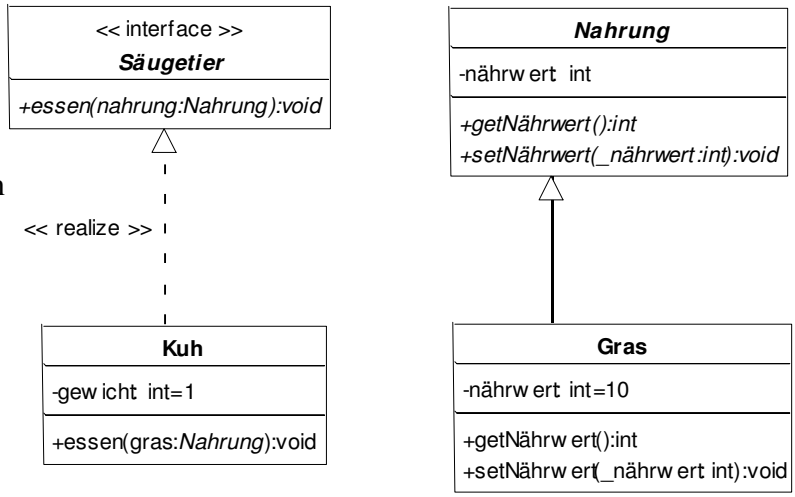

Abbildung 2.1:Säugetierbeispiel UML-Diagramm

In beiden Fällen werden abstrakte Methoden vorgegeben, die erst in der vererbenden Klasse zu implementieren sind: essen(Nahrung) für das Säugetier und get/setNährwert() für die Nahrung. Ein Unterschied ist aber, das die abstrakte Basisklasse Eigenschaften (Attribute) wie zum Beispiel den Integerwert „nährwert“ besitzen darf. In einer Schnittstelle dürfen dagegen außer den Methodensignaturen allenfalls Konstanten definiert werden. Tabelle 2.1 stellt die Unterschiede, wie sie sich im Quelltext manifestieren, gegenüber. Farbig hinterlegt sind die unterschiedlichen Schlüsselwortkombinationen

(interface<-> implements, abstract <-> extends), die den wesentlichen Unterschied in der Formulierung des Quelltextes ausmachen.

Säugetier.java

```
public interface Säugetier {

  public abstract void essen(Nahrung
nahrung);

}
```

Nahrung.java

```
public abstract class Nahrung {

  private int nährwert;

  public abstract int getNährwert();

  public abstract void setNährwert
(int _nährwert);

}
```

Kuh.java

```
public class Kuh implements Säugetier
{

  private int gewicht = 1;

  public void essen(Nahrung gras) {
    gewicht += gras.getNährwert();
    gras.setNährwert(0);
  }

}
```

Gras.java

```
public class Gras extends Nahrung {

  private int nährwert = 10;

  public int getNährwert() {
    return nährwert;
  }

  public void setNährwert(int
_nährwert) {
    nährwert = _nährwert;
  }

}
```

Tabelle 2.1: Säugetierbeispiel Quelltext

Interfaces und Vererbung sind Konzepte, die in javabasierten Frameworks häufig verwendet werden. Durch Beziehungen zwischen den einzelnen Klassen können so komplexe Strukturen entstehen – das „Rahmenwerk“ einer Anwendung.

3 Technologien

In diesem Kapitel wird das Webpublishing-Framework Cocoon 2 vorgestellt und gezeigt, wie es funktioniert. Danach werden die weiteren für das Projekt verwendeten Softwaretechnologien vorgestellt.

3.1 Java

Als Entwicklungsplattform für die javabasierten Programmkomponenten kommt die zur Zeit der Erstellung dieser Diplomarbeit aktuelle Version des Java 2 SDKs für Linux der Firma Sun Microsystems zum Einsatz, die „Java Platform, Standard Edition (J2SE) 1.4.2_01“. Im Gegensatz zur Java Runtime Environment (JRE) enthält das SDK auch einen Compiler um Javaprogramme zu erzeugen und einige hilfreiche Tools für Serverapplikationen. Die „Java Platform, Enterprise Edition (J2EE) ist Suns Referenzimplementation eines Application Servers und wird nicht benötigt.

3.2 Build-Tools

Ant[1] von der Apache Software Foundation ist ein Open-Source-Build-Tool für Java, ähnlich dem `make` unter Unix. Ant selbst ist ein Javaprogramm und kann daher auf verschiedenen Betriebssystemen eingesetzt werden. Es wird zum Kompilieren von vielen Programmen (z. B. Cocoon oder Tomcat) benötigt und ist sehr verbreitet, jede bessere Java-IDE bietet inzwischen die Integration von Ant.

3.3 Frameworks

3.3.1 Cocoon 2

Cocoon ist ein Projekt der Apache Software Foundation, das von Stefano Mazzocchi 1998 zu dem Zweck gestartet wurde eine Web-Applikation für die Dokumentation der

1 Das FAQ erklärt den Namen folgendermaßen: According to Ant's original author, James Duncan Davidson, the name is an acronym for "Another Neat Tool". Later explanations go along the lines of "ants do an extremely good job at building things", or "ants are very small and can carry a weight dozens of times their own" - describing what Ant is intended to be.

Javaprojekte der Foundation zu entwickeln. Das zu diesem Zeitpunkt vorliegende Dokumentationsmaterial war ein Sammelsurium aus HTML-Dokumenten aus unterschiedlichsten Quellen, von unterschiedlichster Qualität und Form. Dieses Informationsmaterial in eine einheitliche Form bringen zu wollen veranlasste ihn dazu eine Software zu entwickeln, die den Informationen eine Struktur verpassen sollte.

Mazzocchi erkannte nämlich die große Schwäche von HTML, das zwar in der Lage ist zu beschreiben wie Inhalte angezeigt werden sollen, aber keinerlei semantische Informationen über den Inhalt eines Dokumentes mitbringt. So ist es für einen Menschen, der eine HTML-Seite betrachtet, leicht zu erkennen und abzugrenzen, wo sich ein Newsticker, eine Navigationsleiste, ein Werbebanner eine Überschrift und ein Textabsatz befinden – schaut man sich den HTML-Quellcode einer solchen Seite an, sieht man aber nur eine Anhäufung von Tabellen und verschachtelten Formatierungsanweisungen, die keinen eindeutigen Hinweis auf die Bedeutung ihres Inhalts geben. Ein entscheidender Nachteil solcher HTML-Dokumente ist, dass sie nicht oder nur sehr schwer maschinenlesbar beziehungsweise von Maschinen interpretierbar sind.

Mazzocchi entwickelte mit Cocoon 1 ein Programm, das in der Lage war, Dokumente, deren Inhalt in Form von XML vorlag, mittels der Verarbeitungssprache XSLT in HTML oder ein anderes Format umzuwandeln. Dadurch schaffte er es, den Inhalt eines Dokumentes von den Formatierungselementen zu trennen.

Das Ziel von Cocoon ist es, die unterschiedlichen an der Entwicklung einer Website beteiligten Arbeitsgruppen von den sie nicht betreffenden Aufgaben abzuschirmen, um die Produktivität zu erhöhen. Zum Beispiel ist es uneffektiv, wenn ein Programmierer in einer Anwendung HTML-Code schreiben muss, obwohl er eigentlich nur Inhalte generieren sollte und eigentlich ein Webdesigner für diese Arbeit zuständig wäre.

Cocoon bietet eine Möglichkeit bestimmte „Verträge“ zwischen den Gruppen zu definieren, die als Schnittstelle zur Zusammenarbeit dienen. Das so entstehende Modell aus Verträgen (Contracts) und Anliegen (Concerns) der Arbeitsgruppen ist in Abbildung 3.1 dargestellt. Die Cocoon-Entwickler nennen diese Herangehensweise „Separation of Concerns“ (SoC).

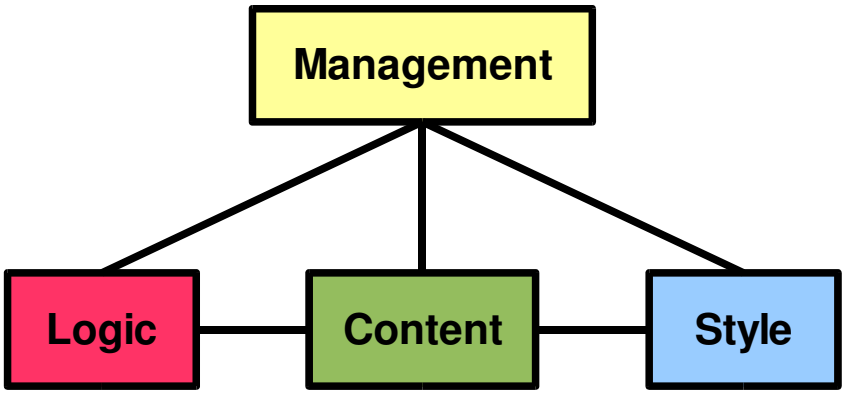

Abbildung 3.1: Das Cocoon-Pyramidenmodell der Verträge

Bei Cocoon 1, das hauptsächlich eine Transformation von XML mittels XSLT vorsieht, gibt es jedoch einige Einschränkungen. Es basiert auf der DOM-Level-1-API. Diese API realisiert XML-Zugriffe durch Javaobjekte, die ein XML-Dokument repräsentieren. Dieser eigentlich intuitive Ansatz birgt leider Schwächen in der Verarbeitungsgeschwindigkeit und Effizienz der Speichernutzung, da für jedes Dokument neue Objekte angelegt und wieder gelöscht (Garbage Collection) werden müssen. Dieser Vorgang ist aber gerade eine der größten Schwachstellen aktueller Javalaufzeitumgebungen. Die Garbage Collection, also das Aufräumen ungenutzter Speicherbereiche, erfolgt automatisch in unregelmäßigen Abständen. Die automatische Speicherverwaltung ist zwar sehr nützlich, aber hinsichtlich der Performance alles andere als optimal.

Cocoon 1 benutzt „Reactor Patterns“, um die Programmkomponenten miteinander zu verbinden. Deshalb muss man „Reaction Instructions“ innerhalb des XML-Dokumentes platzieren. Obwohl dies bequem ist, verursacht es Probleme beim Verwalten von hoch dynamischen Websites.

Cocoon 2 geht bei der XML-API einen neuen Weg. DOM, als Objektstruktur, wird von der ereignisbasierten Struktur SAX ersetzt. Dies optimiert die Verarbeitungsgeschwindigkeit und Effizienz der Speichernutzung, die Skalierbarkeit und die Bearbeitung von sehr großen XML-Dokumenten. Die „Reactor Patterns“ werden abgelöst durch ein pipelineartiges Verarbeitungsmodell (siehe Abbildung3.2).

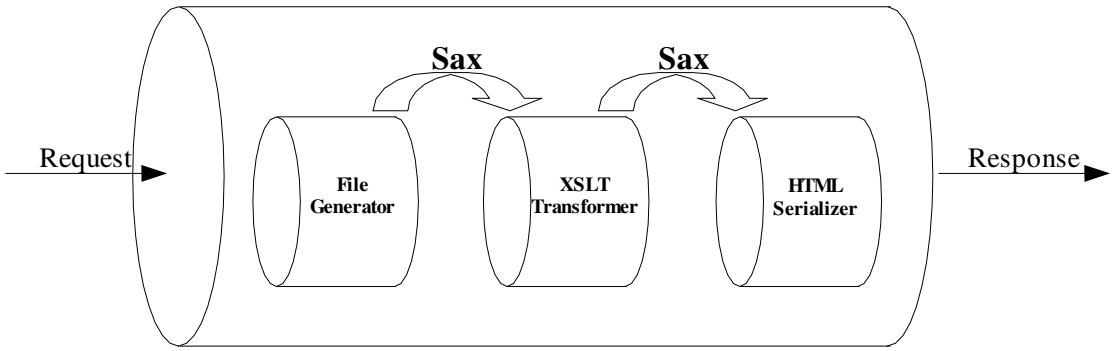

Abbildung 3.2: Die ereignisbasierte Pipeline-Architektur von Cocoon 2

Die Cocoon-2-Pipeline schaltet folgende Komponenten hintereinander:

- **Generator**
  Ein Generator erstellt eine XML-Struktur aus einer Eingabequelle (Datei, Verzeichnis, Stream...).
- **Transformer (optional)**
  Ein Transformer wandelt eine Input-XML-Struktur in eine andere XML-Struktur um. Ein häufig benutzter Transformer ist XSLT, andere sind z.B. SQL Transformer, LOG Transformer.
- **Aggregator (optional)**
  Ein Aggregator macht im Prinzip das Gleiche wie ein Generator, nur dass er als Eingabequelle mehrere XML-Strukturen erwartet. Diese kann er zu einer neuen XML-Struktur zusammenfügen.
- **Serializer**
  Ein Serializer wandelt eine Input-XML-Struktur in eine andere Datenstruktur (Binary oder Character-Stream), die nicht notwendigerweise etwas mit XML zu tun hat, z. B. HTML, Text, PDF, SVG, PNG, JPEG.

[ACo03]

3.3.2 Apache Avalon Phoenix

Das Avalon-Projekt[2] ist ein Framework und eine Sammlung von Komponenten zur einfachen Entwicklung von Serveranwendungen mithilfe der Programmiersprache Java. Phoenix[3] ist ein Micro-Kernel, der auf dem Avalon Framework aufbaut. Es stellt viele

2 Avalon auch Avalun genannt, ist eine aus der Artussage stammende im Nebel liegende Insel bei England, die sich nur den Eingeweihten zeigt. Die Uneingeweihten erreichen nur Glastonbury.

3 In der ägyptischen Mythologie ist der Phoenix ein heiliger Vogel. Am Ende seines Lebens baut er ein Nest, setzt sich hinein und verbrennt. Wenn die Flammen verlöschen, bleibt ein Ei zurück, aus dem nach kurzer Zeit ein neuer Phoenix schlüpft.

Möglichkeiten, die Umgebung von Serveranwendungen zu verwalten zur Verfügung. Dazu gehören zum Beispiel Logmanagement, Classloading, Threadmanagement und Sicherheitsmechanismen. Der Kern des Reporting-Frameworks xReporter läuft in diesem Framework.

3.3.3 Apache Tomcat

Tomcat[4] ist eine Servlet-Engine, das heißt in Tomcat werden Java-Servlets ausgeführt. Tomcat wird zum Beispiel benötigt, um Cocoon 2 auszuführen. Es lässt sich aber auch jede andere Servlet Engine, die die Servlet API 2.2 erfüllt für Cocoon 2 verwenden.

3.3.4 Cocoondev.org xReporter

xReporter ist ein webbasiertes Datenbank-Reporting-Framework. Mit xReporter kann man Datenbankberichte (Reports) ohne besondere Programmierkenntnisse erstellen.

xReporter wurde in Java geschrieben und basiert auf vielen Open-Source-Projekten der Apache Foundation, wie zum Beispiel Avalon, Cocoon und Tomcat. Die Präsentationsebene von xReporter profitiert von modernen Webtechnologien wie XML und XSLT, da sie auf Apache Cocoon aufsetzt. Die Kernkomponente zur Erzeugung der Berichtsdaten setzt auf das Framework Apache Avalon Phoenix auf.

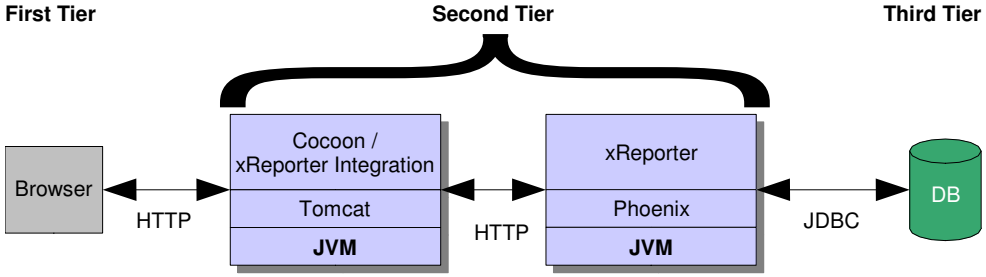

Abbildung 3.3: Die Architektur von xReporter

Abbildung 3.3 zeigt die einzelnen Komponenten von xReporter und die Kommunikationswege untereinander.

4 engl. „Kater“. Ein Posting in der Tomcat-User-Liste erörtert die Namensgebung: „He (James Duncan Davidson) figured that since most open source projects had O'Reilly books about them that he should name it after an animal. Essentially he was thinking of an animal that would go on the cover of an O'Reilly book. He came up with "Tomcat" since the animal represented something that could take care of itself and fend for itself.“

3.3.5 Cocoondev.org Fins

Fins[5] ist eine Komponente für Cocoon, die es ermöglicht aus XML-Daten Diagramme in verschiedenen Bildformaten (PNG, JPEG, SVG) zu rendern. Sie basiert auf den Javaklassen von JFreeChart, einer Javaklassenbibliothek für Diagramme. JFreeChart nutzt die grafischen Primitivoperationen des Abstract-Window-Toolkits (AWT) von Java, um Diagrammelemente zu zeichnen.

3.4 Das Benutzerkonzept von Linux

Um Serveranwendungen auf einem Computer der an ein Netzwerk angeschlossen ist mit dem notwendigen Augenmerk auf Betriebssicherheit einrichten zu können, ist es wichtig, sich mit dem Sicherheitskonzept des Betriebssystems vertraut zu machen.

Das wichtigste und grundlegendste Sicherheitssystem ist das Benutzerkonzept mit seinen Zugriffsrechten auf Dateien und Objekte.

Linux erlaubt es mehreren Anwendern gleichzeitig auf dem System zu arbeiten. Prinzipiell gibt es zwei verschiedene Benutzergruppen, die sich am System anmelden können: Administratoren (Root) und einfache Benutzer. Daneben gibt es noch einige andere vom System intern verwendete Benutzer, die zum Beispiel dazu genutzt werden Systemdienste mit bestimmten Benutzerrechten auszuführen.

Der Benutzer Root ist nach der Installation auf jedem System vorhanden und besitzt privilegierte Rechte. Das bedeutet, dass dieser Benutzer das System warten und konfigurieren darf. Root besitzt damit leider auch alle Rechte, die benötigt werden, um ein Linuxsystem wissentlich oder unwissentlich zu zerstören, anzugreifen oder kaputtzukonfigurieren. Um dies zu vermeiden, sollte man für die tägliche Arbeit sowie für das Starten von Diensten unbedingt einen „normalen“ Benutzer mit eingeschränkten Rechten anlegen. Für den Picos-Report-Server wird deshalb ein generischer Benutzer mit dem Namen „picosuser“ eingerichtet. Alle Prozesse, die zum Report-Server gehören, werden im Kontext und mit den eingeschränkten Rechten dieses Benutzers gestartet.

5 englisch „Flossen“

3.5 Systemdienste unter Red Hat Linux

Um den Report-Server produktiv einsetzen zu können, muss er als Hintergrunddienst installiert werden. Um dies für den Report-Server zu erreichen, muss er dem System als Dienst bekannt gemacht werden. Den Start von Diensten erledigt unter Unix-Systemen der `init`-Prozess. `Init` ist das erste Programm, das vom Kernel gestartet wird, es kümmert sich neben den Diensten auch um die Basiskonfiguration des Systems.

Der Start von Hintergrunddiensten erfolgt auch bei Red Hat mittels des System-V-Init-Prozesses.

Um Javaprogramme als Hintergrundprozesse unter Linux zu starten, benötigt man zusätzliche Software. Ein sehr beliebtes Programm für diesen Zweck ist der „Java Service Wrapper“.

3.5.1 Java-Service-Wrapper

Ein Problem der Javaframeworks Phoenix und Cocoon ist, dass sie vom Benutzer als Anwendung manuell gestartet werden müssen. Phoenix lässt sich außerdem nur im Vordergrund ausführen und blockiert dabei die Konsole. Wünschenswert wäre es aber, die beiden Prozesse als Dienst im Hintergrund (unter Linux „Daemon“[6] genannt) auszuführen, da dies in der Natur dieser benutzeroberflächenlosen Anwendungen liegt.

Um dies zu ermöglichen, ist der Java-Service-Wrapper sehr gut geeignet. Er unterstützt verschiedene Möglichkeiten, eine Javaanwendung als Dienst zu integrieren.

3.6 Dateiübertragungsprotokolle

Eine wesentliche Aufgabe der zu entwickelnden Anwendung ist die Übertragung von Daten über eine Fernverbindung. Für eine solche Kommunikation zwischen entfernten Computern muss eine geeignete Sprache (Protokoll) gewählt werden. Moderne Betriebssysteme besitzen bereits umfangreiche Eigenschaften, um über Netzwerke mit einander zu kommunizieren. Dabei nutzen sie in der Regel das frei zugängliche Netzwerk-

6 Die Herkunft des Begriffes ist nicht ganz sicher, die Programmierer des Project MAC am Massachusetts Institute of Technology sollen ihn geprägt haben. Demnach war der erste Daemon ein automatisches Bandsicherungsprogramm, der Begriff Daemon die Kurzform für „Disk And Execution MONitor“.

protokoll TCP/IP. Dieses lässt sich zusammen mit allen erdenklichen physikalischen Datenleitungen verwenden, darunter zum Beispiel analoge Telefonleitung, ISDN, Twisted-Pair-Kabel, Koaxialkabel, Glasfaser, sogar Luft (W-LAN). Zwei Computer können über eine Punkt-zu-Punkt-Verbindung (z. B. mit Modems) ein Minimalnetzwerk mittels TCP/IP aufbauen, über das sie miteinander kommunizieren können. Weil die Daten als Textdateien übertragen werden sollen, muss ein Dateiübertragungsprotokoll, das auf TCP/IP basiert, eingesetzt werden. Dateiübertragungen über Betriebssystemgrenzen hinweg lassen sich am besten mit dem FTP-Protokoll durchführen.

3.6.1 FileZilla FTP-Server

Windows 2000 in der Version für Arbeitsplätze bringt keinen FTP-Server mit, diese Funktionalität gehört zu Microsofts Internet-Information-Server, der in die Servervarianten von Windows 2000 integriert ist. Eine Alternative aus dem Open-Source-Bereich ist FileZilla-Server. FileZilla ist ein quelloffener FTP-Server für Windowsbetriebssysteme. Er kann als Windowsdienst installiert werden, sodass er nach dem Start des Betriebssystems im Hintergrund läuft, ohne dass ein Benutzer eingeloggt sein muss. Der merkwürdige Name entstand in Anlehnung an das etwas bekanntere Open-Source-Projekt „Mozilla“ - ein Webbrowser für diverse Betriebssysteme.

3.6.2 Pavuk FTP-Client

Die Datenübertragung vom WinCC-PC zum Linuxserver wird mittels eines Skripts realisiert, dass ein Programm zum Spiegeln von FTP-Servern steuert. Das für solche Aufgaben unter Linux häufig benutzte GNU Wget ist in diesem Fall leider nicht geeignet, da es keine Dateien auf dem entfernten FTP-Rechner nach dem Download löschen kann. Auf der Wget Homepage findet man aber einen Link auf die Seite von Lachlan Cranswick, der das Tool Pavuk[7] von Stefan Ondrejicka als mögliche Alternative zu Wget verlinkt (http://www.idata.sk/~ondrej/pavuk/about.html); mit dem Hinweis, dass Pavuk auch in der Lage ist Dateien zu löschen.

7 slowakisch „Spinne“

3.7 PPP-Verbindungen

Die Kommunikationsvorgänge zwischen den Computern, also WinCC-PC auf der einen Seite und Datenbankserver auf der anderen Seite müssen über irgendeine Form von Netzwerkprotokoll erfolgen. Für die Übertragung von Dateien ist das in diesem Fall FTP. Mit FTP alleine lässt sich allerdings noch keine Verbindung zwischen zwei Computern aufbauen, denn FTP ist nur eine von mehreren Schichten aufeinander aufbauender Protokolle, dem so genannten Protokollstapel, dargestellt im Schichtenmodell für TCP/IP. Jeder der Schichten kommt dabei eine bestimmte Aufgabe zu. Erst eine Kombination aus jeweils einem Protokoll aus allen vier Schichten des Schichtenmodells ergibt das Grundgerüst für eine funktionsfähige Anwendung. Das für die Herstellung einer TCP/IP-Verbindung über Wählleitungen benötigte Protokoll heißt PPP (Point-to-Point-Protocol). Tabelle 3.1 zeigt die Einordnung von FTP und PPP in die Schichten des TCP/IP-Protokollstapels.

Schicht	*Beispielprotokolle*
Anwendung	**FTP**, SMTP, Telnet
Transport	TCP, UDP
Netzwerk	IP, ICMP
	ARP, RARP
Netzzugang	**PPP**, Ethernet, Token Ring, FDDI, W-LAN
	Modem, ISDN, Twisted-Pair-Kabel, Glasfaser, Luft

Tabelle 3.1: TCP/IP-Schichtenmodell

PPP regelt die Authentifizierung des sich einwählenden Benutzers (optional sogar die des Servers) und die Vergabe der IP-Adressen. PPP ist ein standardisiertes Protokoll, das mit vielen Betriebssystemen benutzbar ist. Es existieren jedoch einige leicht unterschiedliche Dialekte, die einen reibungslosen Verbindungsaufbau erschweren können.

3.7.1 Eingehende PPP-Verbindung unter Windows

Der Dienst, der unter Windows-NT Betriebssystemen Wählverbindungen verwaltet, nennt sich RAS (Remote Access Service).

Unter Windows lassen sich eingehenden Wählverbindungen ganz unkompliziert mithilfe eines Assistenten einrichten. Eine solche Verbindung wird an ein physikalisches Gerät gebunden (zum Beispiel Modem oder ISDN-Adapter), mit Benutzerrechten geschützt und mit einer Politik zur Vergabe von IP-Adressen versehen (automatische Vergabe über DHCP oder Angabe eines IP-Bereichs).

Auf die Auswahl des Authentifizierungsprotokolls oder ähnlicher Feinheiten hat man leider keinen Einfluss. Das birgt insbesondere dann Probleme in sich, wenn man versucht sich mit einem anderen Betriebssystem als Microsoft Windows einzuwählen.

3.7.2 Ausgehende PPP-Verbindung mit pppd

Auf dem Server-PC muss für jede Schmiedeanlage, deren Daten importiert werden sollen, eine eigene PPP-Verbindung eingerichtet werden. Linux baut solche Verbindungen mit pppd auf. Zur Erleichterung der Konfiguration einer solchen Wählverbindung stehen unter Linux verschiedene Hilfsprogramme zur Verfügung. Dazu gehören reine Textmodusprogramme (WvDial) und Programme mit grafischen Oberflächen (kppp, rp3, linuxconf). Diese Programme bieten den Vorteil der einfachen Konfiguration und des bequemen Verbindungsaufbaus.

Leider ist der Authentifizierungsprozess, mit dem sich der einwählende Rechner am Remote-Rechner anmeldet, nicht einheitlich standardisiert – es gibt eine Vielzahl von Authentifizierungsprotokollen (PAP, CHAP) von denen es wiederum diverse nicht standardisierte Varianten gibt (MS-CHAP, MS-CHAP V2). Der Kreativität eines Einwahl-Providers, welche Protokolle er in welcher Form unterstützt, sind keine Grenzen gesetzt.

Ein weiteres Problem ist die Automatisierbarkeit: Die PPP-Verbindung soll in einem Skript für die Datenübertragung aufgebaut und wieder beendet werden. Dazu muss ein Befehl existieren, der sich ohne grafische Oberfläche aber mit Parametern ausführen lässt, sodass nach Wunsch eine bestimmte Nummer gewählt werden kann.

Das Wählprogramm WvDial erfüllt im Prinzip die oben aufgeführten Anforderungen, jedoch ergaben sich im Testbetrieb mit der Einwahl auf dem Windows 2000 RAS-Server Probleme bei der Authentifizierung. An dieser Stelle kommt man mit WvDial nicht

weiter, da man keinen Einfluss auf die Auswahl und Durchführung des Authentifizierungsmechanismus hat. Was übrig bleibt, ist die Konfiguration des PPP-Daemons von Hand.

3.8 Datenkompression

Die Übertragung von Textdateien über eine analoge Telefonleitung klappt zwar prinzipiell problemlos, kann aber sehr lange dauern, wenn sich viele Daten angesammelt haben. Ein Nachteil der Textdateien ist, dass sie in der Regel einen höheren Anteil redundanter Daten im Vergleich zu Binärdateien enthalten. Die redundanten Daten müssen aber leider mitübertragen werden. Um diesen Overhead zu verringern, werden alle Textdateien in eine komprimierte Archivdatei verschoben. Ein Test mit 742 Beispieldateien aus der Laufzeitdatenbank ergab dabei eine Größenreduktion der Archivdatei auf ungefähr 14% der Originalgröße aller archivierten Dateien. Somit verringert sich auch die Übertragungszeit um (mindestens) 86%.

Als Archivformat bietet sich Zip an, da es sowohl in der Windows- als auch in der Unixwelt etabliert ist. Ein sehr bekanntes freies Zipprogramm ist Info-Zip. Info-Zip ist ein Kommandozeilenprogramm, das sich sehr schön in automatische Prozesse einbinden lässt. Die Einbindung des Packprogramms[8] erfolgt auf der Windowsseite durch einen entsprechenden Aufruf am Ende des täglichen Exportvorgangs, auf der Linuxseite durch einen Aufruf des Entpackprogramms innerhalb des Dateiübertragungsskripts.

8 Datenkompressionsprogramme, die Archivdateien erzeugen können werden oft auch als „Archivprogramme“, „Packer“ oder deutsch: „Packprogramme“ bezeichnet.

4 Installation und Konfiguration

In diesem Kapitel wird die Installation und Inbetriebnahme der verwendeten Software beschrieben. Dazu gehören die Komponenten für die Applikation auf der Seite des Servers: Java 2 SDK, Frameworks, Skripts, sowie auf der Seite des Clients ein FTP-Server und eine „Eingehende PPP-Verbindung“. Nicht näher beschrieben werden die vorbereitende Installation des Server-Betriebssystems Red Hat Linux 9.0 und die Installation von Oracle9i für Linux, die für dieses Projekt von Fachleuten im Hause vorgenommen wurden.

Die hier beschriebenen Installations- und Konfigurationsarbeiten dienen im Wesentlichen dazu, das Betriebssystem und insbesondere die verwendeten Frameworks in einen für die Entwicklung und den Betrieb einer Reporting-Applikation erforderlichen Zustand zu versetzen. Die Entwicklung der eigentlichen Anwendung auf Basis der Frameworks gehört nicht zu diesem Themenkreis und wird im Kapitel „Softwareentwicklung“ erläutert.

Alle folgenden Programme, die auf dem Server installiert werden müssen, werden in das Verzeichnis `/usr/local` installiert, auf das nur Root Schreibzugriff hat. Dieses Verzeichnis ist laut [Kof00] in Linuxsystemen für Anwendungen und Dateien vorgesehen, die nicht unmittelbar zur Linuxdistribution gehören oder später installiert wurden. Eine Ausnahme bilden diejenigen Komponenten, die für die Ausführung im Kontext eines normalen Benutzers in dessen Benutzerverzeichnis installiert werden müssen. Dazu gehören insbesondere Teile der Frameworks Phoenix und Tomcat.

Die Unterkapitel im Einzelnen:

- 4.1 Anlegen eines Benutzers
- 4.2 Java 2 SDK
- 4.3 Apache Ant
- 4.4 Apache Avalon Phoenix
- 4.5 Apache Tomcat

- 4.6 Cocoondev.org Fins
- 4.7 Apache Cocoon
- 4.8 Cocoondev.org xReporter
- 4.9 Java Service Wrapper
- 4.10 Pavuk
-
-
-
-
-

4.1 Anlegen eines Benutzers

Die Anmeldung eines neuen Benutzers geschieht am einfachsten über die Konfigurationsprogramme der jeweiligen Distribution. Unter Red Hat Linux ist das der „Benutzer-Verwalter“.

Um diesen zu öffnen, muss man gegebenenfalls erst die grafische Oberfläche mit `startx` starten. Daraufhin erscheint der Gnome-Desktop. Man klickt nun im Hauptmenü auf „Systemeinstellungen/Benutzer und Gruppen“. Ist man nicht als root angemeldet, muss man jetzt noch das Root-Passwort eingeben. Über die Funktion „Benutzer hinzufügen“ legt man bequem einen neuen Benutzer an, in diesem Fall mit den Eigenschaften:

Benutzername: picosuser
Vollständiger Name: generischer Picos Benutzer
Passwort: ********

Alle übrigen Werte belässt man bei den Standardeinstellungen.

Sofern in den folgenden Installationsanweisungen ein Kommando als root ausgeführt werden muss, wird darauf explizit hingewiesen. Alle übrigen Kommandos werden als *picosuser* ausgeführt.

4.2 Java 2 SDK

Zunächst muss man die entsprechende Version des SDK von Suns Website herun-

terladen. Man findet die Downloadseiten unter folgender URL:

http://java.sun.com/j2se/downloads.html

Für Linux bietet Sun zwei Installationsvarianten an:

- Linux self-extracting file
  Damit lässt sich das SDK in einem beliebigen Verzeichnis installieren, auch von Benutzern ohne Root-Rechte. Es kann zusätzlich zu einer systemweiten Javaversion installiert werden.
- Linux RPM in self-extracting file
  Diese selbstextrahierende Datei enthält ein Installationspaket im für Red Hat üblichen RPM-Format. Das SDK lässt sich nur von Root und nur im Standard-Java-Verzeichnis installieren. Vorteil dieser Variante ist, dass sich das Paket über die RPM-Funktionalität leicht updaten und deinstallieren lässt.

Aufgrund der RPM-Funktionalität empfiehlt es sich, das RPM-Installationspaket zu verwenden. Nach dem Download in ein beliebiges Verzeichnis muss man die Datei erst einmal mit den Rechten zur Ausführung ausstatten:

```
chmod a+x j2sdk-1_4_2_01-linux-i586-rpm.bin
```

Danach kann man die Datei ausführen mit

```
./j2sdk-1_4_2_01-linux-i586-rpm.bin
```

Man beachte „./“ am Anfang des Befehls. Diese Pfadangabe ist nötig, falls der aktuelle Pfad nicht in der PATH-Umgebungsvariable enthalten ist. Es folgt eine Lizenzvereinbarung, der man zustimmen muss, dann wird das Installationspaket `j2sdk-1_4_2_01-linux-i586.rpm` im aktuellen Verzeichnis erzeugt.

Ab hier muss man mit Root-Rechten weiterarbeiten, indem man `su` und auf Nachfrage das Super-User-Passwort eingibt.

Danach startet man rpm, um das SDK zu installieren:

```
rpm -iv j2sdk-1_4_2_01-linux-i586.rpm
```

Das SDK wird somit im Verzeichnis `/usr/java/j2sdk1.4.2_01` installiert.

Jetzt muss nur noch die PATH-Variable um das Binary-Verzeichnis des Java-SDK ergänzt werden. Um diese Einstellung global vorzunehmen, das heißt für alle Benutzer,

muss ein Eintrag in der `/etc/profile` erfolgen. Gleichzeitig empfiehlt es sich die Variable JAVA_HOME zu setzen, die von vielen Javaprogrammen benötigt wird.

```
# /etc/profile

...

# Path for Java 2 SDK
PATH=$PATH:/usr/java/j2sdk1.4.2_01/bin

# Java Homedir
export JAVA_HOME=/usr/java/j2sdk1.4.2_01
```

/etc/profile

Danach ist Java installiert und kann von jedem Verzeichnis aus mit dem Befehl `java` ausgeführt werden. Ein kleiner Test: `java -version` sollte die Versionsnummer und ggf. den Hersteller des SDKs anzeigen.

4.3 Apache Ant

Die binäre Distribution des Build-Tools Ant liegt in der Version 1.5.4 vor. Nach dem Download des TAR-Archivs von der Ant-Homepage http://ant.apache.org/bindownload.cgi bzw. von einer der Mirrorsites muss dieses mit `tar` entpackt werden.

```
tar -xcf apache-ant-1.5.4-bin.tar.gz -C /usr/local
```

Auch für Ant wird das Binary-Verzeichnis dem Pfad hinzugefügt und eine Umgebungsvariable ANT_HOME eingerichtet.

```
# /etc/profile

...

# Path for Java 2 SDK
PATH=$PATH:/usr/java/j2sdk1.4.2_01/bin:/usr/local/apache-ant-1.5.4/bin
...

# Ant Homedir
export ANT_HOME=/usr/local/apache-ant-1.5.4
```

/etc/profile

Jetzt ist Ant einsatzbereit und kann mit `ant [options] [target]` aufgerufen werden.

4.4 Apache Avalon Phoenix

Man bekommt Phoenix als kompilierte Binärdateien hier:

http://www.apache.de/dist/avalon/phoenix/binaries/

Die Zipdatei entpackt man als picosuser ins Homeverzeichnis mit

```
unzip phoenix-4.0.4-bin.zip -d ~
```

Dann macht man die Skripts ausführbar:

```
cd /usr/local/phoenix-4.0.4/bin
chmod a+x *.sh
```

Damit ist Phoenix bereits installiert, aber noch nicht einsatzbereit. Denn um diesem Framework auch einen Sinn zu geben, muss man noch den Phoenix-Teil von xReporter als Phoenix-Anwendung installieren. Dieser Schritt wird im nächsten Abschnitt beschrieben.

4.5 Apache Tomcat

Tomcat wird, wie die meisten anderen Javaapplikationen der Apache Foundation auch in verschiedenen Archivformaten angeboten. Diese unterscheiden sich aber nicht in ihrem Inhalt. Die einzigen plattformspezifischen Dateien sind die Startskripts, die aber in jedem Archivformat komplett enthalten sind. Die Distribution als Zip-Datei lässt sich sowohl mit Windows als auch mit Linux verwenden.

Das Entpacken der Zip-Datei erfolgt unter Linux mit

```
unzip tomcat-4.1.27.zip -d /usr/local
```

Man muss nun die Skripts im Binary-Verzeichnis von Tomcat manuell mit den Rechten zum Ausführen versehen:

```
cd /usr/local/jakarta-tomcat-4.1.27/bin
chmod a+x *.sh
```

Tomcat kann eine Verbindung zu einem Webserver aufbauen oder im Standalone-Modus HTTP-Verbindungen bedienen. Der Standalone-Modus ist für die Entwicklung der Applikation sehr gut geeignet.

Um Konflikte mit dem XDB-Server von Oracle9i zu vermeiden, der wie Tomcat auf Port 8080 horcht, sollte man die Standardeinstellung von Tomcat auf einen anderen Port ändern.

Die Verbindungseinstellungen findet man in der Konfigurationsdatei `<tomcat home>/conf/server.xml`.

In Zeile 93 ist der Port 8080 im Attribut „port“ des Standard HTTP-Connectors definiert.

```
91:<!-- Define a non-SSL Coyote HTTP/1.1 Connector on port 8080 -->
92:   <Connector className="org.apache.coyote.tomcat4.CoyoteConnector"
93:     port="8080" minProcessors="5" maxProcessors="75"
94:     enableLookups="true" redirectPort="8443"
95:     acceptCount="100" debug="0" connectionTimeout="20000"
96:     useURIValidationHack="false" disableUploadTimeout="true" />
```

<tomcat home>/conf/server.xml

Wenn man diesen Wert zum Beispiel auf 8887 ändert, gibt es keine Probleme mehr. Mit dem Befehl `./startup.sh` im Verzeichnis „bin“ kann man Tomcat jetzt im Hintergrund starten. Mit einem Webbrowser sollte sich jetzt die Tomcat-Testseite unter http://<rechnername>:8887 bzw. http://localhost:8887 aufrufen lassen. Mit `./shutdown.sh` lässt sich Tomcat wieder beenden.

4.5.1 Tomcat als Benutzerinstallation

Um Tomcat in einer etwas sichereren Umgebung laufen zu lassen, ist es besser ihn so zu installieren, dass er auch von einem nicht privilegierten Benutzer ausgeführt werden kann. In der oben beschriebenen Standardinstallation darf Tomcat nämlich nur als root ausgeführt werden.

Um dies erreichen zu können, bietet Tomcat die Möglichkeit dynamische Programmteile in einem separaten Verzeichnis unterzubringen, siehe Tomcat-Dokumentation [Apa02]. Dieses Verzeichnis muss in der Umgebungsvariablen CATALINA_BASE angegeben werden. Diese kann man zum Beispiel in der Konfigurationsdatei `/home/picosuser/.bashrc` exportieren:

```
export CATALINA_BASE=~/tomcat
```

/home/picosuser/.bashrc

Bei diesem Beispiel wird das Unterverzeichnis „tomcat“ im Home-Verzeichnis des Benutzers „picosuser“ als CATALINA_BASE verwendet, wenn dieser Benutzer Tomcat startet.

Zu den Standardverzeichnissen von Tomcat, die in diesem Verzeichnis untergebracht werden gehören:

- conf
- logs
- temp
- webapps

Den Inhalt der Verzeichnisse „conf“ und „webapps“ muss man aus dem Tomcat-Installationsverzeichnis kopieren. In „logs“ und „temp“ werden nur von Tomcat selbst erzeugte Dateien abgelegt. Diese Verzeichnisse muss man einfach nur anlegen - wenn sie fehlen, funktioniert Tomcat nicht ordnungsgemäß.

4.6 Cocoondev.org Fins

Die Fins-Homepage befindet sich hier: http://www.cocoondev.org/projects/fins.html.

Zusätzliche oder optionale Komponenten von Cocoon werden seit Version 2.1 als „Blocks“ vor der Kompilierung des Frameworks eingebunden. Um eine eigene Konfiguration zu erstellen, die festlegt, welche Blocks eingebunden werden sollen, muss man die Datei `blocks.properties` aus dem Cocoon-Verzeichnis (/usr/local/cocoon-2.1.1) in die Datei `local.blocks.properties` umkopieren und diese dann bearbeiten. Wie man zu dem Cocoon-Verzeichnis kommt, ist Abschnitt „Apache Cocoon“ weiter unten beschrieben.

Zur Installation geht man wie folgt vor:

Man lädt die Quelltexte von http://www.sidimar.ipzs.it/fins011/docs/download/fins-0.1.1.zip herunter. Man entpackt die Zipdatei in `src/blocks` unterhalb des Cocoon-Verzeichnisses.

```
unzip fins-0.1.1.zip -d /usr/local/cocoon-2.1.1/src/blocks
```

Dann legt man ein Verzeichnis lib im Fins-Verzeichnis an.

```
mkdir /usr/local/cocoon-2.1.1/src/blocks/fins/lib
```

In dieses kopiert man die Bibliotheken `jfreechart-0.9.11.jar` und `jcommon-0.8.6.jar` von der JFreeChart Webseite (http://www.jfree.org/jfreechart/index.html). Auch die Bibliothek `batik-all-1.5b5.jar` aus `src/blocks/batik/lib` muss in dieses Verzeichnis kopiert werden.

Zu der Datei cocoon-2.1.1/lib/jars.xml, die alle Bibliotheken von Cocoon beschreibt, muss man folgende Zeilen innerhalb des <jars>-Elementes hinzufügen:

```
<!-- files for Fins -->
<file>
  <title>JFreeChart</title>
  <description>Charting package</description>
  <used-by>Fins block</used-by>
  <lib>fins/lib/jfreechart-0.9.11.jar</lib>
  <homepage>http://www.jfree.org/jfreechart/index.html</homepage>
</file>

<file>
  <title>JCommon</title>
  <description>Utility library used by JFreeChart</description>
  <used-by>Fins block</used-by>
  <lib>fins/lib/jcommon-0.8.6.jar</lib>
  <homepage>http://www.jfree.org/jcommon/index.html</homepage>
</file>

<file>
  <title>Batik</title>
  <description>
  Batik is a Java based toolkit for applications which handle images
in
  the Scalable Vector Graphics (SVG) format for various purposes, such
as
  viewing, generation or manipulation.
  </description>
  <used-by>Fins block</used-by>
  <lib>fins/lib/batik-all-1.5.jar</lib>
  <homepage>http://xml.apache.org/batik/</homepage>
</file>
```

cocoon-2.1.1/lib/jars.xml

Zu der Datei cocoon-2.1.1/gump.xml, die die Abhängigkeiten der Komponenten von

Cocoon beschreibt, muss man folgende Zeilen innerhalb des <module>-Elementes hinzufügen:

```
<!-- Fins project -->
<project name="cocoon-block-fins" status="unstable">
  <package>it.ipzs.charts</package>

  <ant target="gump-block">
  <property name="block-name" value="fins"/>
  <property name="version" value="@@DATE@@"/>
  </ant>

  <depend project="cocoon" inherit="all"/>
  <depend project="xml-batik"/>

  <work nested="tools/anttasks"/>
  <home nested="build/cocoon-@@DATE@@"/>

  <jar name="blocks/fins-block.jar"/>

  <nag from="Gump" to="cocoon-dev@xml.apache.org"/>
</project>
```

cocoon-2.1.1/gump.xml

Jetzt sollte man noch die Datei `local.blocks.properties` um folgende Zeilen ergänzen:

```
# Fins block
#exclude.block.fins=true
```

Datei cocoon-2.1.1/local.blocks.properties

Hier gilt das Nummernzeichen als Einleitung eines Zeilenkommentars, das heißt der Schlüssel exclude.blocks.cocoon wird nicht auf true gesetzt und erhält damit den Standardwert false, was wiederum bedeutet, dass Fins mitkompiliert wird. Man könnte `local.blocks.properties` also auch einfach unangetastet lassen.

Schlussendlich ersetzt man das <jar>-Element in der Datei `cocoon-2.1.1/tools/src/blocks-build.xsl` noch mit diesem:

```
<!-- jar element modified for Fins -->
<jar
jarfile="{string('${build.blocks}')}/{$block-name}-block.jar"
basedir="{string('${build.blocks}')}/{$block-name}/dest"/>
```

cocoon-2.1.1/tools/src/blocks-build.xsl

Jetzt kann man mit der Installation von Cocoon fortfahren. Die mitgelieferten Beispieldiagramme kann man sich unter http://localhost:8888/samples/fins/index.html anschauen, sobald man Cocoon erfolgreich gestartet hat.

4.7 Apache Cocoon

Die genaue Vorgehensweise für Beschaffung und Installation von Cocoon wird in [Apa03] beschrieben. Cocoon kann man unter der URL http://cocoon.apache.org/mirror.cgi downloaden. Cocoon wird seit der Version 2.1 nur noch als Quellcode veröffentlicht. Man muss das Archiv daher nicht nur entpacken

```
unzip cocoon-latest-src.zip -d /usr/local
```

sondern auch noch kompilieren. Vor dem Kompilieren sollte man die benötigten Blocks ausgewählt haben, zum Beispiel Fins (siehe oben). Die für Cocoon mit Fins und xReporter benötigten Blocks sind authentication-fw, batik, fop, poi, session-fw und axis. Von den Schlüsseln der anderen Blocks kann man das Kommentarzeichen (#) entfernen. Dadurch werden sie exkludiert.

```
cd /usr/local/cocoon-2.1.1
chmod a+x *.sh
./build.sh
```

Daraufhin wird Cocoon als Web-Applikation im Unterverzeichnis `build/webapp` erstellt. Jetzt muss man ein Verzeichnis im Webapps-Verzeichnis von Tomcat erstellen, in das die Web-Applikation dann kopiert wird.

```
cd /usr/local/jakarta-tomcat-4.1.27/webapps
mkdir cocoon
cd /usr/local/cocoon-2.1.1/build/webapp
cp --recursive * /usr/local/jakarta-tomcat-4.1.27/webapps/cocoon
```

Zum Testen startet man Tomcat und kann, wenn alles geklappt hat, die Cocoon Startseite unter http://<rechnername>:8888/cocoon/ bewundern.

4.8 Cocoondev.org xReporter

Die letzte Releaseversion 1.2 von xReporter funktioniert leider noch nicht zusammen

mit Cocoon 2.1. Auf die Nachfrage im xReporter-Entwicklerforum antwortete Bruno Dumon, einer der Entwickler, dass aber die derzeitige Entwicklerversion Cocoon 2.1 unterstützte und stabil sie, da daran nicht mehr viel verändert würde.

Der Zugriff auf die Entwicklerversion (=CVS-Version) erfolgt über das Versionsverwaltungssystem CVS – mit diesem hat man lesenden Zugriff auf den Quellcode von xReporter. Alternativ dazu kann man auch ein regelmäßig neu erzeugtes Archiv mit dem kompletten Quellcode unter `http://xreporter.cocoondev.org/tarball/` herunterladen. Dort findet man ein Verzeichnis mit mehreren aufeinander folgenden „Schnappschüssen“ des aktuellen Quellcodes – die jeweils aktuellste Version befindet sich am Ende der Liste. Zum Beispiel ist die Version vom 01.10.2003 in der Datei `xreporter-cvs-20031001.tgz` zu finden.

4.8.1 Installation

In diesem Kapitel wird die Installation von xReporter in Form einer lauffähigen Testinstallation beschrieben. Die Vorgehensweise richtet sich dabei grob nach der Installationsanleitung für xReporter 1.2.1, die im Downloadpaket von xReporter enthalten ist. Die Anleitungen für die Installation der unterschiedlichen xReporter-Versionen finden sich auch in der Dokumentation [Out03].

Die Installation erfolgt als picosuser mit

```
mkdir xreporter
cd ~/xreporter
tar -xzf xreporter-cvs-20031001.tgz
```

Die Zusammenhänge zwischen den einzelnen Frameworks und xReporter sind recht kompliziert und unübersichtlich, deshalb folgt an dieser Stelle eine Übersicht über das xReporter-Verzeichnis und die Bedeutung der darin enthaltenen Daten.

Die im xReporter-Verzeichnis erzeugten Unterverzeichnisse enthalten folgende Daten:

Unterverzeichnis	*Inhalt*
`testconf`	eine Beispielkonfiguration für xReporter mit Testdaten und Testreports auf Basis einer MySQL-Datenbank (wird nicht benötigt)
`xreporter`	Die Phoenix-Komponente von xReporter.
`bin`	Skripts zum Kompilieren des Quellcodes und zum Kopieren nach Phoenix (Deployment[1])
`lib`	Binäre Javabibliotheken von Drittanbietern
`src`	Javaquellcode von xReporter und Dokumentation im Cocoon-Documentation-Book-Format (XML)
`xreporter-cocoon`	Die Cocoon-Komponente von xReporter.
`bin`	Skripts zum Kompilieren des Quellcodes und zum Kopieren nach Cocoon (Deployment)
`lib`	Binäre Javabibliotheken von Drittanbietern
`src`	Javaquellcode von xReporter-Cocoon und allgemeine Ressourcendateien (Stylesheets, Bilder, Übersetzungstabellen)

Tabelle 4.1: xReporter-Installationsverzeichnis

Im Prinzip funktioniert die Installation von xReporter also folgendermaßen:

- man erstellt eine Konfiguration, d. h. Reportdefinitionen, Datenbanktreiber und -einstellungen, Datenquellen, Datentypen und Autorisierung in einem Unterverzeichnis von xReporter (z. B. testconf)
- man kompiliert die Phoenix-Komponente und kopiert sie ins Anwendungsverzeichnis von Phoenix (Phoenix-Deployment)
- man kompiliert die Cocoon-Komponente und kopiert sie ins Webapp-Verzeichnis von Cocoon (Cocoon-Deployment)

Die Kompilations- und Deploymentvorgänge werden jeweils von einem Skript durchgeführt, das man im gewünschten Konfigurationsverzeichnis ausführt. Wenn man dann Phoenix und Tomcat/Cocoon ausführt, integrieren sich die xReporter-Komponenten

1 engl. „Stationierung“, „Aufstellung“, „Errichtung“. Software-Deployment ist ein etwas umfassenderer Begriff als Softwareinstallation, er beinhaltet zum Beispiel auch Konfiguration, Anpassung, Installation, Update, Aktivierung und Deinstallation. Auch die Verbreitung von Software über große Netzwerke nimmt eine immer größere Bedeutung in diesem Themenkreis ein.

mehr oder weniger von alleine in ihr Framework und werden dann automatisch gestartet.

4.8.2 Start mit einer neuen Konfiguration

Zur Speicherung von Reports und zur Verwaltung von Benutzerrechten muss man zunächst einige Tabellen in der Datenbank anlegen. Die dafür benötigten SQL-Skripts findet man in der Installationsanleitung **~/**`xreporter/src/documentation/content/xdocs/en/installv121.xml`

In der Datei `~/.bashrc` erweitert man den Path um die Binärverzeichnisse von xReporter. Dadurch kann man die später benötigten Befehle „xreporter“ und „xreporter-cocoon“ in jedem Verzeichnis ausführen.

```
export PATH=$PATH:./:~/xreporter/xreporter/bin:~/xreporter/xreporter-cocoon/bin
```

~/.bashrc

Nun geht es an die Erstellung der Konfiguration. Als Erstes muss man ein Verzeichnis anlegen, in dem die Konfiguration gespeichert werden soll:

```
mkdir ~/xreporter/picosconf
```

Danach erzeugt der Befehl „seed“ im Verzeichnis eine Standardkonfiguration:

```
cd ~/xreporter/picosconf
xreporter seed
```

Dabei wird das Programm auch schon kompiliert und in `picosconf/build` abgelegt. Dieses Build könnte man schon als Anwendung in Phoenix installieren, das würde aber keinen Sinn machen, da ja noch keine Reports erstellt wurden und keine Datenquellen eingerichtet wurden.

Der nächste Schritt dient der Einrichtung einer Datenbankverbindung. Man öffnet die XML-Datei

`~/xreporter/picosconf/sarconf/config.xml` mit einem Texteditor und sucht nach folgenden Einträgen:

```
<connection-params>
  <url>jdbc:mysql://localhost/xreporteradmin</url>
```

```
  <user>xreporteradmin</user>
  <password>xreporteradmin</password>
</connection-params>
```

~/xreporter/picosconf/sarconf/config.xml

Dieses XML-Tag kommt kommt viermal in der Datei vor, einmal zur Konfiguration der Datenbankverbindung für den Zugriff auf die Benutzertabellen (users, roles, customers, userprops), einmal für die Autorisierungstabellen (accesscontrol, authcodes), einmal für den User-entry-store (userentrystore) und einmal für gespeicherte Reports (reportstore). Diese Tabellen sind exakt die, die man im ersten Schritt in der Datenbank angelegt hat. Diese Einstellungen sind an die tatsächliche Datenbank anzupassen, zum Beispiel mit:

```
<connection-params>
  <url>jdbc:oracle:thin:@//127.0.0.1:1521/picosrs</url>
  <user>picosuser</user>
  <password>picosuser</password>
</connection-params>
```

~/xreporter/picosconf/sarconf/config.xml

Wie man sieht, wird in der URL eine JDBC-Verbindung zu einer Oracle-Datenbank mit Namen „picosrs“ beschrieben. Da man dafür auch einen Oracle-JDBC-Treiber braucht, muss man diesen auch zu xReporter hinzufügen. Ein entsprechender Treiber findet sich nach der Installation der Oracle-Clientsoftware für Windows in `c:\orant9\jdbc\lib\ojdbc14.jar`.

Diese Datei kopiert man in `~/xreporter/picosconf/lib` auf dem Linux-PC. Damit xReporter die Datei auch lädt, muss man den Abschnitt <drivers-to-load> in der `config.xml` um den Eintrag

```
<driver class="oracle.jdbc.OracleDriver"
location="@conf.home@/lib/ojdbc14.jar"/>
```

~/xreporter/picosconf/sarconf/config.xml

erweitern.

Ein Bericht (report) in xReporter benötigt eine so genannte Datenquelle (datasource), um auf Berichtsdaten zuzugreifen. Diese Datenquellen werden in `picosconf/conf/da-`

tasources.xml konfiguriert. Für die Daten des Picos-Report-Servers werden folgende Einstellungen benötigt.

```
<datasource id="picosrs-ds">
  <name>Picos Report Server Datenquelle</name>
  <description> (Oracle9i Datenbank)</description>
  <sortcode1></sortcode1>
  <sortcode2></sortcode2>
  <supported-types>
    <type>picosdata</type>
  </supported-types>
  <connection-params>
    <url>jdbc:oracle:thin:@//128.1.1.160:1521/picosrs</url>
    <user>picosuser</user>
    <password>picosuser</password>
    <!-- For an empty username and/or password,
         remove the tag(s) completely -->
  </connection-params>
</datasource>
```

~/xreporter/picosconf/conf/datasources.xml

Die hier selbst definierte Datenquelle „picosrs-ds“ wird nun noch mit einem Autorisierungscode verknüpft, der es dem Benutzer mit diesem Code ermöglicht, auf diese Datenquelle zuzugreifen. Welchem Benutzer welcher Autorisierungscode zugewiesen ist, wird in der Datenbanktabelle „authcodes“ konfiguriert. Die Zuweisung von Autorisierungscodes zu Datenquellen erfolgt in ~/xreporter/picosconf/conf/authorisationcodes.xml.

Hier fügt man einfach dem Element <datasource-authorisation-code id="sample-auth-code"> die Datenquelle mit der id „picosrs-ds“ hinzu.

```
<?xml version="1.0"?>
<!-- This file defines the data source authorisation codes. -->
<!-- An authorisation code specifies to which data sources a
     user will have access. -->
<datasource-authorisation-codes>

  <datasource-authorisation-code id="sample-auth-code">
    <!-- Here comes a list of datasource id's -->
    <datasource id="sample-ds"/>
    <datasource id="picosrs-ds"/>
  </datasource-authorisation-code>
```

```
</datasource-authorisation-codes>
```

~/xreporter/picosconf/conf/authorisationcodes.xml

Um die Konfiguration testen zu können, passt man jetzt den Testreport ~/`xreporter/picosconf/reports/firstreport.xml` ein wenig an.

Der Datenquellentyp wird geändert auf „picosdata“, wie in der Datenquellenkonfiguration (datasources.xml) angegeben:

```
<required-datasource-type>picosdata</required-datasource-type>
```

~/xreporter/picosconf/reports/firstreport.xml

Der SQL-Befehl wird auf eine einfache Abfrage des Systemdatums eingestellt:

```
<literal>SELECT SYSDATE FROM DUAL</literal>
```

Dann ersetzt man die Column-Elemente mit passenden Einträgen:

```
<columns>
  <column id="field1" field="SYSDATE">
    <title>Oracle Systemdatum</title>
    <type base="string"/>
  </column>
</columns>
```

~/xreporter/picosconf/reports/firstreport.xml

Dadurch erhält man einen Report, der das aktuelle Systemdatum der Oracle-Datenbank anzeigt.

4.8.3 Phoenix Deployment

Die Installation in Phoenix (Deployment) und Cocoon wird mit den Skripts in den Bin-Verzeichnissen durchgeführt. Damit die xReporter Skripts wissen, wo Cocoon und Phoenix im System installiert sind, muss man diese Pfadangaben in der Datei `~/xreporter/picosconf/build.properties` eintragen.

Um diese zu erstellen, kopiert man einfach die vorhandene `build.properties.sample` um. Danach ändert man folgende Werte:

```
phoenix.home=/home/picosuser/phoenix-4.0.4
cocoon.home=/home/picosuser/tomcat/webapps/cocoon
```

~/xreporter/picosconf/build.properties

Um xReporter in Phoenix zu installieren, führt man dieses Kommando aus:

```
cd ~/xreporter/picosconf
xreporter deploy
```

Danach kann man testen, ob Phoenix sich starten lässt:

```
~/phoenix-4.0.4/bin/run.sh
```

Daraufhin sollte folgende Ausgabe auf dem Bildschirm erscheinen:

```
Using PHOENIX_HOME:   /home/picosuser/phoenix-4.0.4
Using PHOENIX_TMPDIR: /home/picosuser/phoenix-4.0.4/temp
Using JAVA_HOME:      /usr/java/j2sdk1.4.2_01
Running Phoenix:

Phoenix 4.0.4

xReporter HTTP server started
```

Phoenix ist gestartet und wartet auf Verbindungen, beenden lässt sich das Programm mit der Tastenkombination STRG+C.

4.8.4 Cocoon Deployment

Bevor die xReporter-Komponenten für Cocoon installiert werden können, müssen in Tomcat und Cocoon einige zusätzliche Einstellungen vorgenommen werden. xReporter unterstützt Benutzerauthentifizierung – allerdings führt es diese nicht selbst durch, sondern überlässt sie dem Servlet Container, in diesem Fall Tomcat. Dazu muss in Tomcat ein Realm erstellt werden. In der ~/tomcat/conf/server.xml fügt man dem Element <Engine> folgendes Realm-Element hinzu:

```
<Realm  className="org.apache.catalina.realm.JDBCRealm"
       driverName="oracle.jdbc.OracleDriver"
    connectionURL="jdbc:oracle:thin:@//127.0.0.1:1521/picosrs"
   connectionName="picosuser" connectionPassword="picosuser"
        userTable="users" userNameCol="username"
userCredCol="password"
    userRoleTable="roles" roleNameCol="role" />
```

~/tomcat/conf/server.xml

Andere Realm-Elemente müssen auskommentiert oder entfernt werden, sonst funktioniert die Authentifizierung nicht. Damit auch Tomcat auf den Datenbanktreiber

zugreifen kann, muss man diesen (als root) in das Verzeichnis

`/usr/local/jakarta-tomcat-4.1.27/server/lib` kopieren.

Zur Definition welche Bereiche geschützt sein sollen, fügt man „Security-Constraints“ in die Konfiguration des Cocoon Servlets `~/tomcat/webapps/cocoon/WEB-INF/web.xml` ein. Das Element <login-config> beschreibt, mit welcher Methode die Authentifizierung erfolgen soll (HTML-Formular) und welche Seite als Loginseite angezeigt werden soll, wenn ein (noch) nicht authentifizierter Benutzer eine Seite aus dem geschützten Bereich anfordert.

```
<!-- xReporter security constraints -->
<security-constraint>
  <web-resource-collection>
    <web-resource-name>XReporter</web-resource-name>
    <url-pattern>/xreporter/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>*</role-name>
  </auth-constraint>
</security-constraint>

<login-config>
  <auth-method>FORM</auth-method>
  <realm-name>XReporter protected area</realm-name>
  <form-login-config>
    <form-login-page>/xreporter-login</form-login-page>
    <form-error-page>/xreporter-loginfailed</form-error-page>
  </form-login-config>
</login-config>
```

~/tomcat/webapps/cocoon/WEB-INF/web.xml

Um die xReporter Login- und Logoutseiten in Cocoon einzubinden, müssen die entsprechenden Einträge in der Sitemap gemacht werden. Die Sitemap enthält alle Informationen die Cocoon benötigt, um Antworten aus den Anfragen, die es bekommt, zu generieren. Anders als bei einem Webserver, der statische Dateien auf Anfrage aus einem real existierenden Verzeichnis bereitstellt, bietet Cocoon im Prinzip eine virtuelle Verzeichnisstruktur feil. Cocoon kann aus einer Anfrage mithilfe der Sitemap ermitteln, welche Arbeitsschritte zur Bedienung der Anfrage notwendig sind und somit das gewünschte Dokument generieren. Die Sitemap wird in `~/tomcat/webapps/cocoon/si-`

`temap.xmap` konfiguriert. Vor der Main-Pipeline fügt man folgendes Pipeline-Element hinzu:

```
<!-- xReporter login pipelines. These should be put outside the
protected area, since the login
   pages themselves obviously should not be protected. -->
<map:pipeline>
  <map:match pattern="xreporter-login">
    <map:read
      src="xreporter/resources/skins/{xreporterdefaults:skin}/
html/login.html"
      mime-type="text/html"/>
  </map:match>

  <map:match pattern="xreporter-loginfailed">
    <map:read
      src="xreporter/resources/skins/{xreporterdefaults:skin}/
html/loginfailed.html"
      mime-type="text/html"/>
  </map:match>

  <!-- matcher to point to skin-specific images. Note that this
circumvents the security,
       but for skin files this normally doesn't matter -->
  <map:match pattern="xreporter-login/images/*.gif">
    <map:read
      src="xreporter/resources/skins/{xreporterdefaults:skin}/images/
{1}.gif"
      mime-type="image/gif"/>
  </map:match>

  <map:match pattern="xreporter-login/images/*.png">
    <map:read
      src="xreporter/resources/skins/{xreporterdefaults:skin}/images/
{1}.png"
      mime-type="image/png"/>
  </map:match>

  <!-- matcher to point to skin-specific css files. Note that this
circumvents the security,
       but for skin files this normally doesn't matter -->
  <map:match pattern="xreporter-login/css/*.css">
    <map:read
      src="xreporter/resources/skins/{xreporterdefaults:skin}/css/{1}.
css"
      mime-type="text/css"/>
  </map:match>

  <!-- The logout matcher is also put outside the protected area, to
```

```
prevent an
       endless login loop: suppose the authentication-session has
expired, and then
       the user follows the logout link; this would then cause the
login page to be
       shown, and after login the user would be redirected to the
logout page, which
       would again cause the login page to be shown, etc -->

  <map:match pattern="xreporter-logout">
    <map:act type="session-invalidate"/>
    <map:act type="request">
      <map:parameter name="parameters" value="true"/>
      <map:redirect-to uri="{goto}"/>
    </map:act>
  </map:match>
</map:pipeline>
```

~/tomcat/webapps/cocoon/sitemap.xmap

Der Logoutvorgang benutzt die Session-Invalidate-Action von Cocoon, die standardmäßig nicht aktiviert ist. Deshalb muss dies in der Sitemap innerhalb des Elements <map:actions> hinzugefügt werden.

```
<!-- xReporter actions -->
<map:action logger="sitemap.action" name="session-invalidate"
src="org.apache.cocoon.acting.SessionInvalidatorAction"/>
```

~/tomcat/webapps/cocoon/sitemap.xmap

Die Komponente, die die Kommunikation zwischen Cocoon und xReporter (Phoenix) durchführt wird in `~/tomcat/webapps/cocoon/WEB-INF/cocoon.xconf` innerhalb des <cocoon> Elements deklariert:

```
<!-- xReporter component for communication between xReporter and
Cocoon -->
<component role="org.outerj.xreporter.client.XReporterClient"
   class="org.outerj.xreporter.client.XReporterClientImpl">
  <host>localhost</host>
  <port>8888</port>
  <user>picosuser</user>
</component>
```

~/tomcat/webapps/cocoon/WEB-INF/cocoon.xconf

Und dem Abschnitt <input-modules> fügt man noch folgende Zeilen hinzu:

```
<!-- xReporter component-instance defines default-skin -->
<component-instance name="xreporterdefaults"
class="org.apache.cocoon.components.modules.input.DefaultsMetaModule">
  <values>
    <skin>outerthought</skin>
  </values>
</component-instance>
```

~/tomcat/webapps/cocoon/WEB-INF/cocoon.xconf

Nun sind alle Einstellungen vorgenommen und xReporter kann in Cocoon installiert werden. Dafür sorgt der Befehl

```
cd ~/xreporter/picosconf
xreporter-cocoon deploy
```

Zum Testen der Konfiguration startet man Phoenix und Tomcat und besucht die Seite http://localhost:8887/cocoon/xreporter/en-US/datasources. Die Anmeldung erfolgt mit dem Benutzernamen „jef“ und dem Passwort „bigsecret“.

4.9 Java Service Wrapper

Zunächst muss man den Wrapper von der Projekt-Homepage bei Sourceforge herunterladen: http://sourceforge.net/project/showfiles.php?group_id=39428

Das Tar-Archiv `wrapper_linux_3.0.5.tar.gz` entpackt man im Home-Verzeichnis mit

```
cd ~
tar -xzf wrapper_linux_3.0.5.tar.gz
```

Die weitere Installation wird im Folgenden für Phoenix und Tomcat gesondert beschrieben.

4.9.1 Phoenix als Daemon installieren

Als Erstes sind die beiden Dateien `~/wrapper_linux_3.0.5/bin/wrapper` und `~/wrapper_linux_3.0.5/src/bin/sh.script.in` in das Binary-Verzeichnis von Phoenix zu kopieren. Danach benennt man die Skriptdatei so um, dass sie den Namen der Anwendung widerspiegelt, zum Beispiel „phoenix“.

```
mv sh.script.in phoenix
```

Danach bearbeitet man das Skript und setzt die Umgebungsvariablen APP_NAME und APP_LONG_NAME auf sinnvolle Werte.

```
# Application
APP_NAME="Phoenix"
APP_LONG_NAME="xReporter Phoenix"
```

~/phoenix-4.0.4/bin/phoenix

Dann macht man das Skript ausführbar:

```
chmod a+x phoenix
```

Die beiden Programmbibliotheken `~/wrapper_linux_3.0.5/lib/libwrapper.so` und `~/wrapper_linux_3.0.5/lib/wrapper.jar` sind in das Lib-Verzeichnis von Phoenix zu kopieren. Die Konfigurationsdatei `wrapper.conf` ist bei Phoenix schon vorhanden und kann mit nur einer kleinen Änderung genutzt werden: die Zeile

```
wrapper.java.additional.3=-Djava.security.policy=jar:file:phoenix-
loader.jar!/META-INF/java.policy
```

~/phoenix-4.0.4/conf/wrapper.conf

muss mit # auskommentiert werden.

Ein Test mit Ausgabe eventueller Fehlermeldungen auf der Konsole kann jetzt mit

```
phoenix console
```

durchgeführt werden. Daraufhin sollte folgende Ausgabe am Bildschirm erscheinen:

```
Running xReporter Phoenix...
wrapper  | --> Wrapper Started as Console
wrapper  | Launching a JVM...
jvm 1    | Wrapper (Version 3.0.5)
jvm 1    |
jvm 1    |
jvm 1    | Phoenix 4.0.4
jvm 1    |
jvm 1    | xReporter HTTP server started
```

Das Programm lässt sich mit der Tastenkombination STRG+C wieder beenden.

4.9.2 Tomcat/Cocoon als Daemon installieren

Auch hier sind die beiden Dateien `~/wrapper_linux_3.0.5/bin/wrapper` und `~/wrapper_linux_3.0.5/src/bin/sh.script.in` in das Binary-Verzeichnis von

Tomcat zu kopieren. Dazu legt man das Verzeichnis im persönlichen Tomcat-Verzeichnis an:

```
mkdir ~/tomcat/bin
```

Danach benennt man wieder die Skriptdatei so um, dass sie den Namen der Anwendung widerspiegelt, zum Beispiel „tomcat“.

```
mv sh.script.in tomcat
```

Danach bearbeitet man das Skript und setzt die Umgebungsvariablen APP_NAME und APP_LONG_NAME auf sinnvolle Werte.

```
# Application
APP_NAME="Tomcat"
APP_LONG_NAME="Tomcat/Cocoon"
```

/usr/local/jakarta-tomcat-4.1.27/bin/tomcat

Dann macht man das Skript ausführbar:

```
chmod a+x tomcat
```

Die Bibliotheken `libwrapper.so` und `wrapper.jar` kopiert man in das Verzeichnis `/usr/local/jakarta-tomcat-4.1.27/common/lib`.

Tomcat bringt leider keine eigene wrapper.conf mit, deshalb kopiert man die Vorlage `~/wrapper_linux_3.0.5/src/conf/wrapper.conf.in` nach `~/tomcat/conf/wrapper.conf`.

Diese Konfigurationsdatei muss nun noch bearbeitet werden. Als Erstes muss die Klasse angegeben werden, die die JVM beim Start ausführen soll.

```
wrapper.java.mainclass=org.tanukisoftware.wrapper.WrapperStartStopApp
```

~/tomcat/conf/wrapper.conf

Alle Javaarchive, die beim Booten in den Klassenpfad aufgenommen werden sollen, werden in der Eigenschaft „wrapper.java.classpath“ beschrieben.

```
wrapper.java.classpath.1=/usr/local/jakarta-tomcat-
4.1.27/common/lib/wrapper.jar
wrapper.java.classpath.2=/usr/java/j2sdk1.4.2_01/lib/tools.jar
wrapper.java.classpath.3=/usr/local/jakarta-tomcat-
4.1.27/bin/bootstrap.jar
```

Die Lage der Bibliothek `libwrapper.so` wird dem Wrapper bekannt gemacht.

```
# Java Library Path (location of Wrapper.DLL or libwrapper.so)
wrapper.java.library.path.1=/usr/local/jakarta-tomcat-
4.1.27/common/lib
```

Alle Parameter, die normalerweise von Tomcats Startskripts ermittelt und über Kommandozeilenoptionen an die JVM übergeben werden, müssen in den „zusätzlichen Parametern“ konfiguriert werden. Sehr wichtig für die Verwendung von Fins unter Linux ist dabei der Parameter 5. Dieser schaltet die Java Virtual Machine in den so genannten „Headless Mode“. Dadurch wird verhindert, dass die JVM zum Rendering von Grafiken mithilfe der AWT-Klassen einen laufenden X-Server benötigt. Der X-Server ist eine Komponente von Linux, die zur Darstellung von grafischen Benutzeroberflächen genutzt wird. Seit der Javaversion 1.4 können die AWT-Klassen im Headless Mode auch ohne einen laufenden X-Server Grafiken berechnen. Da der Linuxrechner nur Hintergrundprozesse ausführen soll (Datenbank, Servlets usw.) ist ein laufender X-Server nicht notwendig, ja sogar eher als unnötiges Sicherheitsrisiko anzusehen, weil er einen nicht wirklich benötigten Netzwerkdienst bereitstellt.

```
# Java Additional Parameters
wrapper.java.additional.1=-Djava.endorsed.dirs=/usr/local/jakarta-
tomcat-4.1.27/common/endorsed
wrapper.java.additional.2=-Dcatalina.base=/home/picosuser/tomcat
wrapper.java.additional.3=-Dcatalina.home=/usr/local/jakarta-tomcat-
4.1.27
wrapper.java.additional.4=-Djava.io.tmpdir=/home/picosuser/tomcat/temp
# Important for using Fins
wrapper.java.additional.5=-Djava.awt.headless=true
```

Die folgenden Parameter werden von Wrappers Bootklasse an die Main-Methode der ursprünglichen Bootklasse (org.apache.catalina.startup.Bootstrap) von Tomcat übergeben. Dabei gibt es jeweils eine Konfiguration zum Starten (Parameter 1 bis 3) und zum Stoppen (Parameter 4 bis 7). Beide Vorgänge werden nämlich von der gleichen Klasse ausgeführt.

```
# The first application parameter is the name of the class whose main
# method is to be called when the application is launched.  The class
# name is followed by the number of parameters to be passed to its
# main method.  Then comes the actual parameters.
wrapper.app.parameter.1=org.apache.catalina.startup.Bootstrap
```

```
wrapper.app.parameter.2=1
wrapper.app.parameter.3=start

# The start parameters are followed by the name of the class whose
# main method is to be called to stop the application.  The stop class
# name is followed by a flag which controls whether or not the Wrapper
# should wait for all non daemon threads to complete before exiting
# the JVM.
# The flag is followed by the number of parameters to be passed to the
# stop class's main method.  Finally comes the actual parameters.
wrapper.app.parameter.4=org.apache.catalina.startup.Bootstrap
wrapper.app.parameter.5=true
wrapper.app.parameter.6=1
wrapper.app.parameter.7=stop
```

Ein Test mit Ausgabe eventueller Fehlermeldungen auf der Konsole kann jetzt mit

```
tomcat console
```

durchgeführt werden. Daraufhin sollte ungefähr folgende Ausgabe am Bildschirm erscheinen:

```
Running Tomcat/Cocoon...
wrapper  | --> Wrapper Started as Console
wrapper  | Launching a JVM...
jvm 1    | Wrapper (Version 3.0.5)
[...]
```

Die Startseite von Tomcat kann man dann über http://localhost:8887 erreichen.

Das Programm lässt sich mit der Tastenkombination STRG+C wieder beenden.

4.9.3 Die Dienstekonfiguration von Red Hat Linux

Um einen neuen Dienst zu erstellen, muss man ein Skript schreiben, das einem bestimmten Schema entspricht, es muss zum Beispiel Funktionen zum Starten und Stoppen des Dienstes implementieren. Das Skript für den Picos-Report-Server sieht folgendermaßen aus:

```
#!/bin/bash
# picosrs       Starts Picos Report Server (Tomcat + Phoenix Java
server).
# chkconfig: 345 99 12
# description: Tomcat is the server for Java servlet applications.
Phoenix is the server for XML reports.
### BEGIN INIT INFO
# Provides: $picosrs
```

```
### END INIT INFO

# Source function library.
. /etc/init.d/functions

[ -f /home/picosuser/tomcat/bin/tomcat ] || exit 0
[ -f /home/picosuser/phoenix-4.0.4/bin/phoenix ] || exit 0

RETVAL=0

umask 077

start() {
  echo -n $"Starting Picos Report Server: "
  su -l -c "/home/picosuser/tomcat/bin/tomcat start" picosuser
  su -l -c "/home/picosuser/phoenix-4.0.4/bin/phoenix start" picosuser
  echo
  return $RETVAL
}
stop() {
  echo -n $"Shutting down Picos Report Server: "
  su -l -c "/home/picosuser/tomcat/bin/tomcat stop" picosuser
  su -l -c "/home/picosuser/phoenix-4.0.4/bin/phoenix stop" picosuser
  echo
  return $RETVAL
}
restart() {
  stop
  start
}
case "$1" in
  start)
        start
        ;;
  stop)
        stop
        ;;
  restart|reload)
        restart
        ;;
  *)
        echo $"Usage: $0 {start|stop|restart}"
        exit 1
esac

exit $?
```

/etc/init.d/picosrs

Das Skript basiert auf dem Beispiel aus [Sos03], einem Skript zur Konfiguration von

Tomcat. Um es zu installieren, muss man es als root in das Verzeichnis `/etc/init.d` kopieren und mit

```
chkconfig --add picosrs
```

in die init-Konfiguration einbinden. Sobald das geschehen ist, kann man (zum Beispiel) mittels der grafischen Oberfläche „Dienste-Konfiguration“ von Red Hat den Status des Dienstes überprüfen, den Dienst konfigurieren, starten und stoppen.

4.10 Pavuk

Nach dem Download des Quellcodes lässt sich das Archiv ins aktuelle Verzeichnis entpacken mit

```
tar -xzf pavuk-0.9pl28.tgz
```

Da OpenSSL auf dem Red Hat System nicht installiert ist und für FTP sowieso nicht benötigt wird, muss es bei der Konfiguration deaktiviert werden.

```
cd pavuk-0.9pl28
./configure --disable-ssl
```

Dann kann man es mit

```
make
su
make install
```

kompilieren und installieren.

5 Softwareentwicklung

6 Die Anwendung: Picos-Report-Server

In diesem Kapitel wird das Konzept der Anwendung und das Zusammenspiel der einzelnen Komponenten beschrieben.

6.1 Architektur

Abbildung 6.1 zeigt einen Überblick über die Architektur des Systems.

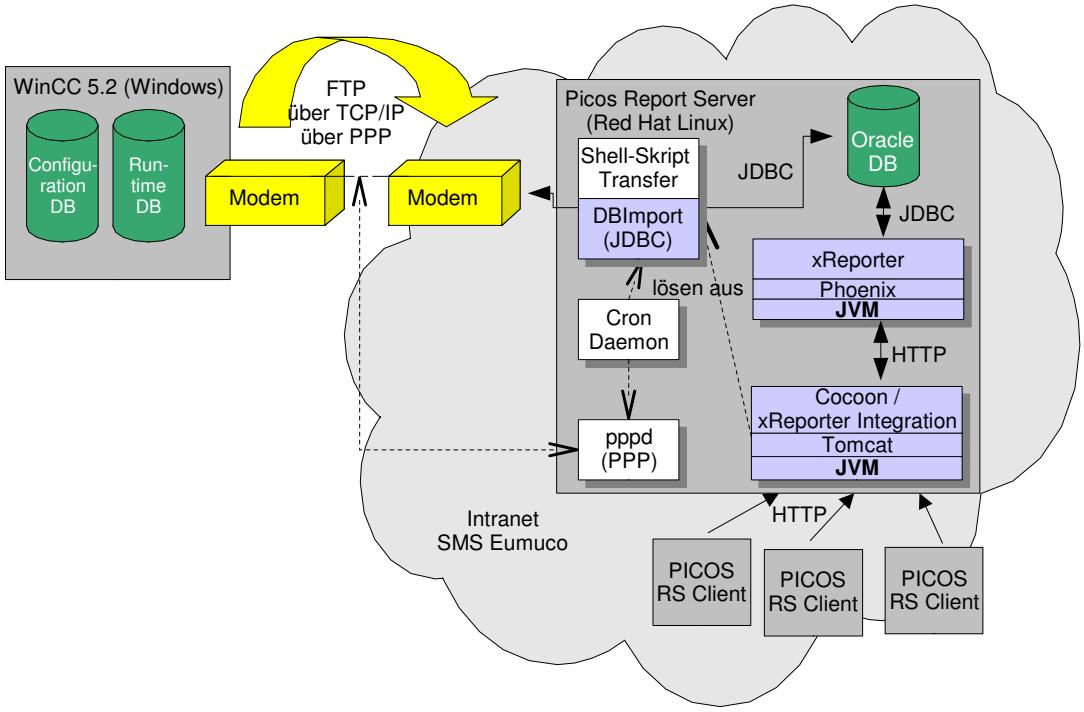

Abbildung 6.1: Die Architektur des Picos-Report-Servers

Jeder graue Kasten stellt einen Computer mit der darauf installierten Software dar. Der Kasten „WinCC 5.2 (Windows)“ auf der linken Seite zeigt das Visualisierungssystem, wie es derzeit auf dem Industrie-PC einer Schmiedeanlage von SMS Eumuco vorzufinden ist. Ein solcher PC ist mit einem Modem verbunden, das zur Herstellung von Netzwerkverbindungen genutzt werden kann. Die Abbildung zeigt eine Konfigurations- und eine Laufzeitdatenbank auf die jedoch nicht direkt zugegriffen werden muss, sondern viel mehr auf täglich exportierte CSV-Dateien mit neuen Laufzeitdaten.

Die gelben Komponenten rechts davon zeigen die an der Datenübertragung beteiligten Geräte und Protokolle. Auf beiden Seiten (Kunde und SMS Eumuco) befindet sich ein PC mit Modem. Der Picos-Report-Server PC nutzt als Betriebssystem allerdings Red Hat Linux und nicht Windows. Eine Netzwerkverbindung wird durch Einwahl des Picos-Report-Servers auf den Windowsrechner mit dem PPP-Protokoll hergestellt. Auf dem Windowsrechner ist dafür eine eingehende PPP-Verbindung konfiguriert und es läuft hier ein FTP-Server.

Um die Dauer der Datenübertragung möglichst kurz zu halten, werden die CSV-Dateien auf Serverseite direkt nach dem Export mithilfe eines kleinen Shell-Skripts in ein komprimiertes Zip-Archiv verschoben.

7 Fazit

Das Ziel dieser Diplomarbeit war die Entwicklung einer webbasierten Anwendung zur Erfassung, Archivierung und Auswertung von Messwerten, Produktions- und Betriebsdaten von Schmiedeanlagen. Die entwickelte Anwendung bietet diese Möglichkeit über ein Intranet-System, womit zugleich jedem Mitarbeiter mit Zugriff auf das Intranet die Anwendung zur Verfügung gestellt werden kann.

Der erste Kernteil der Anwendung – die Übertragung der Daten über eine Fernverbindung – ist zum Zeitpunkt der Abgabe dieser Diplomarbeit bereits in den Regelbetrieb übergegangen. Der Teil der Auswertung der Daten besitzt noch experimentellen Status und ist vor allem durch weitere Reports zu ergänzen. Es wurden jedoch schon einige verwertbare Beispielauswertungen erstellt. Bei der Auswertung ist es jedoch auch denkbar andere Wege zu gehen, beispielsweise durch die Anbindung der in der Firma SMS Eumuco häufig genutzten Reportingsoftware Crystal Reports. Aufgrund der Verwendung der Oracle-Datenbank als Datenspeicher ist eine solche alternative Auswertungsmethode problemlos realisierbar.

Zum Schluss möchte ich mich bei noch den Mitarbeitern der Firma SMS Eumuco für die gute Unterstützung bedanken, insbesondere bei meinen Betreuern Herrn Reiner Rauer, Herrn Bernd Juressen, und Herrn Norbert Gober. Bedanken möchte ich mich außerdem noch bei meiner betreuenden Professorin Frau Prof. Dr. Heide Faeskorn-Woyke.

Anhang A - Literaturverzeichnis

[SMS02] SMS AG. "Geschäftsbericht 2001 der SMS AG", <http://www.sms-eumuco.de/bilder/GB_2001_dt.pdf>, SMS AG, (2002)

[SMS03] SMS AG. "Geschäftsbericht 2002 der SMS AG", <http://www.sms-ag.de/media/GB_2002_deutsch.pdf>, SMS AG, (2003)

[Sie01] Siemens AG, Siemens WinCC 5.1 Online-Hilfe, Nürnberg: Siemens AG, 2001

[Oes01] Oestereich, Bernd, Objektorientierte Softwareentwicklung: Analyse und Design mit der Unified M, 5., München: Oldenbourg, 2001

[Ull03] Ullenboom, Christian, Java ist auch eine Insel, 2. Auflage, Bonn: Galileo Press, 2003

[ACo03] Kumar, Pankaj; Srinivas, Davinum. "Understanding Apache Cocoon", <http://cocoon.apache.org/2.1/userdocs/concepts/index.html>, Apache Software Foundation, (2003)

[Kof00] Kofler, Michael, Linux, 5., München: Addison-Wesley, 2000

[Apa02] The Apache Jakarta Project. "The Tomcat 4 Servlet/JSP Container", <http://jakarta.apache.org/tomcat/tomcat-4.1-doc/index.html>, Apache Software Foundation, (2002)

[Apa03] Kumar, Pankaj; Srinivas, Davanum; Ziegeler, Carsten u. a.. "Cocoon User Documentation", <http://cocoon.apache.org/2.1/userdocs/index.html>, Apache Software Foundation, (2003)

[Out03] Cocoondev.org. "xReporter Documentation", <http://xreporter.cocoondev.org/en/index.html>, Outerthought b.v.b.a. and Schaubroeck n.v., (2003)

[Sos03] Sosnoski, Dennis M.. "Securing Linux for Java services", <http://www-106.ibm.com/developerworks/linux/library/l-secjav.html>, IBM, (2003)

Anhang B – Inhalt der CD-ROM

Diplomarbeit

Dieses Verzeichnis enthält die Diplomarbeit und das Pflichtenheft im PDF-Format.

Picos-Report-Server

Dieses Verzeichnis die Quelltexte und Ressourcen der Anwendung.

Programme

Dieses Verzeichnis enthält die eingesetzten quelloffenen Programme.

Erklärung

Ich erkläre, die von mir vorgelegte Arbeit selbstständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit genutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Mittwoch, 14. Juli 2004